VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analytický nástroj pro systém eLogika

Analytic Tool for eLogika System

2016 Bc. Jakub Gereg

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Gereg**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Analytický nástroj pro systém eLogika
Analytic Tool for eLogika System

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce bude analytický nástroj pro systém eLogika, který bude sloužit k zjišťování chování uživatelů v systému a následnému vyhodnocování.
Na základě výsledků bude systém doporučovat rozmístění jednotlivých prvků v systému eLogika pro rychlejší práci.

Práce bude obsahovat:
1. Analýzu logovaných dat systému eLogika.
2. Navržení metodiky, která bude použita pro analýzu dat.
3. Implementace metodiky a otestování na reálných datech.
4. Návrh dodatečných dat, která se mají logovat pro zefektivnění chování aplikace.

Systém bude umožňovat vizualizaci chování jednotlivých uživatelů v systému.
Aplikační část bude naprogramována v jazyce C#.

Seznam doporučené odborné literatury:

Paul Kimmel: Advanced C# Programming, ISBN: 978-0072224177

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Mgr. Marek Menšík, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2016 .....

Rád bych poděkoval Mgr. Marku Menšíkovi, Ph.D. a Mgr. Pavle Dráždilové, Ph.D. za cenné rady, trpělivost a čas strávený při konzultacích.

Abstrakt

Cílem diplomové práce je vytvoření analytického nástroje pro e-learninový systém eLogika. Budeme se zabývat analýzou chování uživatelů v systému a následovného vyhodnocování.

Záměrem této analýzy je se pokusit nalézt způsoby chování, které je charakteristické pro určité skupiny uživatelů. Na základě analýzy logů systému eLogika, dokážeme doporučovat rozmístění jednotlivých stránek, které bude napomáhat zlepšovat přehlednost a uživatelskou přívětivost systému. Dále dostatečné porozumění analyzovaných dat a případné objevení závislostí, nám může pomoci s predikcí výsledků u budoucích uživatelů systému.

Analytická část je věnována popisu logovaných dat. Jsou zde popsány vybrané metody, které nám pomůžou odhalit a vyhodnotit specifické typy chování. V návrhu si detailněji rozebereme veškeré požadavky, které budeme realizovat ve fázi implementace. Závěr této práce obsahuje výsledky individuálních analýz a jejich zhodnocení.

Klíčová slova: analýza logů, data mining, analytický nástroj, eLogika, e-learning

Abstract

The aim of this thesis is the analytic tool for e-learning system eLogika. Based on user behavior analysis in eLogika system and by following evaluation.

The intention of this analysis is to find types of behavior that are typical for certain groups of users. Based on the analysis of eLogika logs, we can recommend the layout of individual pages, which will help to improve the clarity and user-friendliness of the system. Furthermore, adequate understanding of the analyzed data discovered dependencies, can assist with the prediction of the results for future users of the system.

Analytical part is devoted to describing the logged data. It also described selected methods that will help us identify and evaluate specific types of behavior. The concept will analyze the requirements that we will implement in the implementation phase. The conclusion of this work contains the results of individual analyzes and their evaluation.

Key Words: log analysis, data mining, analytical tool, eLogika, e-learning

Obsah

Seznam použitých zkratek a symbolů

KDD	–	Knowledge Discovery in Database
EDM	–	Educational Data Mining
LMS	–	Learning Management System
REST	–	Representational State Transfer
HTTP	–	Hypertext Transfer Protocol
XML	–	Extensible Markup Language
JSON	–	JavaScript Object Notation
URL	–	Uniform Resource Locator
API	–	Application Programming Interface
IP	–	Internet Protocol
UWP	–	Universal Windows Platform
GUI	–	Graphical user interface
PCL	–	Portable Class Library
DLL	–	Dynamic Link Library
PCA	–	Principal Component Analysis
WPF	–	Windows Presentation Foundation
AHC	–	Agglomerative Hierarchical Clustering
XAML	–	Extensible Application Markup Language
MVVM	–	Model–view–viewmodel
ORM	–	Object-relational mapping
WinRT	–	Windows Runtime
MSDN	–	Microsoft Developer Network
ML	–	Matematická logika

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Tématem diplomové práce je vytvoření analytického nástroje pro eLogiku. Systém eLogika je e-learninový systém, který slouží k řízení studia a výuky přes internet. Hlavním důvodem pro vývoj tohoto nástroje je velké množství dat, které má eLogika k dispozici. Snaha vylepšit systém z uživatelského hlediska a poskytnout dodatečné informace o chování uživatelů, vedla k vytvoření jednotlivých komponent pro analýzu dat v e-learningovém systému eLogika. Tento nástroj má za úkol zpracovávat data týkající se chování uživatelů pomocí analýzy logů přístupů.

Na začátku se seznámíme s podstatou analýzy dat, která zahrnuje její principy, známé postupy a oblasti využití. Na příkladech si nastíníme průběh individuálních kroků v procesu dobývaní znalostí z databází.

V následující kapitole detailněji probereme jednotlivé analytické metody (statistické metody, shlukovou analýzu a asociační pravidla). Pro statistické metody si připomeneme vzorce pro výpočet aritmetického průměru, mediánu, modusu, rozptylu a směrodatné odchylky. Ve shlukové analýze se budeme věnovat metodě K-means, standardizaci dat, redukcí dimenze pomocí PCA a dalším metodám. Asociační pravidla si vysvětlíme na příkladu analýzy nákupního košíku a objasníme si pojem podpora a spolehlivost pravidel.

Ve čtvrté kapitole si ve zkratce popíšeme systém eLogika a zanalyzujeme si data, která máme k dispozici. Podrobněji se budeme věnovat analýze logovaných dat a navrhneme dodatečná data, která by měl systém zaznamenávat pro zefektivnění aplikace.

Další kapitola je věnována samotné analýze a návrhu požadavků, ve které si detailněji rozebereme individuální zadané požadavky a navrhneme si jejich realizaci. Navrhneme si strukturu aplikace a popíšeme si jednotlivé případy užití.

Součástí implementační části je objasnění významu individuálních aplikačních části, použitých knihoven a technologii. Popíšeme si implementaci, komunikace se serverem eLogika, data miningových metod a ostatních použitých algoritmů.

Na konec budou prezentovány výsledky jednotlivých analýz a zhodnotíme jejich vypovídající hodnotu.

2 Analýza dat aneb data mining

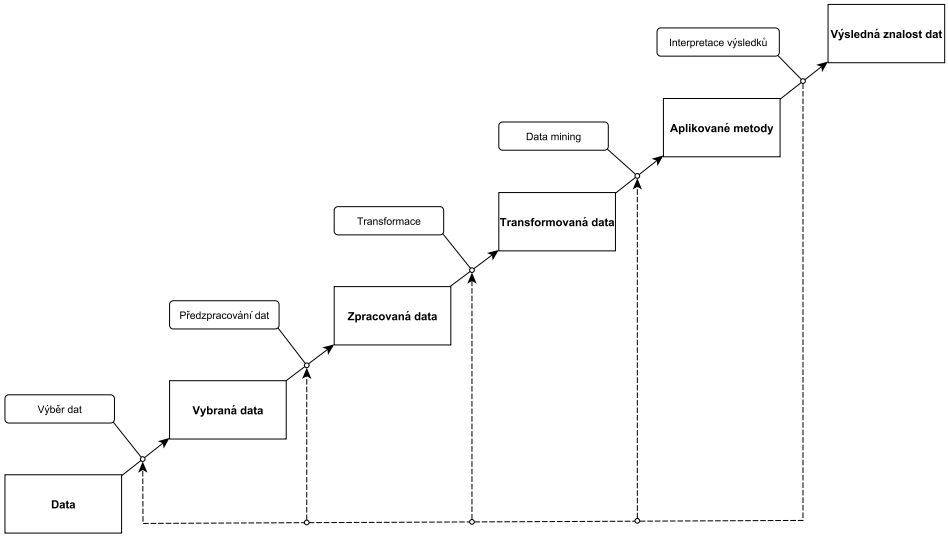
V této kapitole se budeme zabývat analýzou dat. Nejprve bude potřeba si objasnit, jak taková analýza vypadá, a jaké známé metodiky nabízí. Analýza dat, jako taková, dnes není spojena pouze s triviálními statistickými metodami, ale naopak je svázána s rozsáhlými tématy a s širokou škálou přístupů a metodik. Ke statistickým metodám můžeme například přidat metody shlukování, klasifikaci a analýzy vztahů. Takový soubor postupů je v praxi označován za data mining (dolování dat). Více o data miningu v literatuře [1] a [2].

Tyto metodiky jsou klíčové k získávání cenných informací o datech. Ty jsou samy o sobě špatně čitelná a nemají bez důkladnější analýzy téměř žádnou vypovídající hodnotu. Budeme se tedy zabývat obecným využitím Data miningu. Popíšeme si, co vše v datech můžeme nalézt a jaké jsou procesy používané při dolovaní dat z databáze (KDD). V závěru této kapitoly si popíšeme, v jakých oblastech je analýza dat nejčastěji využívána.

2.1 KDD - Dobývání znalostí z databází

Anglické publikace často mylně zaměňují výraz „Knowledge Discovery in Database“, neboli dobývání znalostí z databází, za pojem „data mining“.

Data mining je pouze jeden z několika kroků v procesu KDD. Vezměme si obecný příklad dobývání znalostí z databází zastřešující celou transformaci dat od jejich vzniku až po výstup v procesu KDD. Přehledná ilustrace (viz obr. 8) nastiňuje krok za krokem jednotlivé úkony.



Obrázek 1: Jednotlivé kroky v process KDD

Dobývání znalostí z databází je tedy proces zahrnující výběr, modelování a prohledávání dat. Pomáhá nám také detekovat dříve neznámé vztahy mezi jednotlivými daty. Procesu KDD se detailněji věnují autoři publikací [3] a [4].

Nyní si podrobněji rozepíšeme jednotlivé procesy a uvedeme si názorné příklady vstupních a výstupních dat v jednotlivých krocích.

2.1.1 Výběr vhodných dat

Samotná analýza začíná u výběru vhodných dat. Uvedeme si příklad s e-shopem. Chceme-li například analyzovat, v jakou hodinu lidé nejčastěji navštěvují náš e-shop, zaměříme se na sběr dat obsahující čas a jednotlivé uživatele. U takové analýzy není třeba zacházet nutně do detailů, vystačíme si pouze se základními hodnotami. Naopak analýza obsahu košíku jednotlivých zákazníků již zahrnuje také znalosti všech nabízených produktů. Databáze se nám tímto značně rozšiřuje a sahá až do několika různých oblastí, jako jsou například informace o ceně, počtu zakoupených kusů apod.

Pokud se pouštíme do složitějších analýz s poměrně více datovými zdroji, začleníme do zpracování i postup sjednocení dat. Zprvu chaoticky seskupené informace s velkou obsahovou hodnotou se tím zúží na přehlednější celek, se kterým se bude lépe pracovat. Je tedy vhodné si vybrat pouze data, která jsou nezbytně nutná pro naši analýzu, a tímto redukujeme velké množství nepotřebných dat. Dojde tedy k redukci dat do podoby, která bude relevantní k dané analýze problému.

Příklad: Analýza četnosti nákupu jednotlivých uživatelů v e-shopu v závislosti na čase.

Vstup: V databázové struktuře budou dvě tabulky (viz tab. 1 a 2).

id_uzivatele	jmeno	prijmeni
12	Jan	Novák
13	Petr	Rychlý
14	Karel	Smutný

Tabulka 1: Výběr vhodných dat - seznam uživatelů

id_nakupu	id_uzivatele	cas_nakupu
1	12	01.01.2016 12:21:15
2	13	02.01.2016 11:33:14
3	14	02.01.2016 20:04:12
4	13	04.01.2016 19:33:14

Tabulka 2: Výběr vhodných dat - seznam nákupů

Provedeme sjednocení tabulek do jedné, abychom měli všechny data na jednom místě.

Výstup: Abychom byli schopni analyzovat četnosti nákupů jednotlivých uživatelů e-shopu, v závislosti na čase, bude nám místo dvou předchozích tabulek stačit pouze jedna redukovaná tabulka.

id_nakupu	**id_uzivatele**	**jmeno**	**prijmeni**	**cas_nakupu**
1	12	Jan	Novák	01.01.2016 12:21:15
2	13	Petr	Rychlý	02.01.2016 11:33:14
3	14	Karel	Smutný	02.01.2016 20:04:12
4	13	Petr	Rychlý	04.01.2016 19:33:14

Tabulka 3: Výběr vhodných dat - sjednocení

Nyní už stačí jen spočítat četnosti jednotlivých nákupů a kategorizovat je například na ráno, odpoledne nebo večer.

2.1.2 Předzpracováni dat

Stejně jako redukujeme výběr vhodných dat, i samotný datový obsah se dá vyčistit. Termínem „čištění“ máme na mysli kupříkladu odstranění irelevantních atributů, nepotřebných k vybrané analýze. Dalším krokem může být upravení dat podle vlastních preferencí (změna jmenné konvence atd.).

Příklad: Z tabulky nákupů (viz tab. 2) můžeme ignorovat sloupec *id_nakupu*, který je k zadané analýze nepotřebný. Dále můžeme provést změnu jmenné konvence u sloupce *cas_nakupu* například na *datum_nákupu*, která lépe popisuje daný sloupec.

2.1.3 Transformace

Prvním krokem zahájení analýzy přístupů, bude transformace dat do přijatelného formátu, pro následné zpracování data miningovými metodami. Transformace může zahrnovat různé přepočty, konverze, převody jednotek atd.

Příklad: Převedení četnosti návštěv jednotlivých stránek do kategorii podle četnosti.

Podmínka	Název kategorie	Identifikátor
X > 10	Často	0
10 > X > 3	Méně často	1
3 > X > 0	Skoro vůbec	2

Tabulka 4: Výběr vhodných dat - výčet podmínek pro kategorizaci

Vstup: Tabulka četností navštívených stránek pro každého uživatele.

Uživatel	Hlavní stránka	Druhá stránka	Třetí stránka	Čtvrtá stránka
1	15	10	8	1
2	13	11	2	2

Tabulka 5: Výběr vhodných dat - vstup pro kategorizaci

Výstup: Například uživatel 1, navštívil hlavní stránku celkem patnáctkrát, v podmínce (viz tab. 4) máme uvedeno, pokud navštíví více, než desetkrát zařadíme jej do kategorie s identifikátorem 0 – „často“. Výsledná transformace je zobrazena (viz tab. 6).

Uživatel	Hlavní stránka	Druhá stránka	Třetí stránka	Čtvrtá stránka
1	0	1	1	2
2	0	0	2	2

Tabulka 6: Výběr vhodných dat - kategorizace

2.1.4 Data mining

V tomto kroku dochází k samotné aplikaci data miningových metod, nad již předpřipravenými daty.

Příklad: Použití shlukovací metody K-means.

Vstup: Matice vektorů (viz obr. 2), kde řádky určují individuální vektory V_1 až V_{17}. Sloupce představují dimenze vektorů D_1, D_2 a D_3. Číslo K = 3, které označuje počet výsledných shluků.

$$\begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ ... \\ V_{17} \end{array} \begin{array}{c} \begin{array}{ccc} D_1 & D_2 & D_3 \end{array} \\ \left(\begin{array}{ccc} 1.3 & 2.0 & 1.2 \\ 2.2 & 2.1 & 1.3 \\ 2.1 & 1.2 & 1.1 \\ 3.0 & 1.3 & 2.2 \\ ... & ... & ... \\ 3.7 & 1.6 & 1.1 \end{array} \right) \end{array}$$

Obrázek 2: Příklad - vstupní data pro metodu K-means

Výstup: Metoda K-means (viz kapitola 3.2.2) nám vrátí rozdělení všech vektorů do K shluků.

$$\begin{array}{c} \\ ClusterID \end{array} \begin{array}{c} \begin{array}{cccccc} V_1 & V_2 & V_3 & V_4 & ... & V_{17} \end{array} \\ \left(\begin{array}{cccccc} 1 & 1 & 1 & 2 & ... & 2 \end{array} \right) \end{array}$$

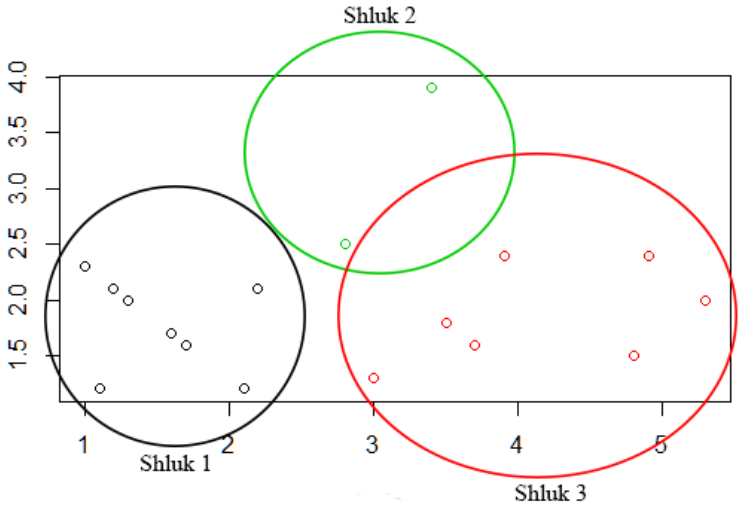
2.1.5 Interpretace výsledků

Pod interpretací výsledků bude reprezentováno finální získání výstupů z jednotlivých data miningových metod, které vhodně prezentujeme uživateli pomocí tabulek, grafů a jiných interaktivních ilustrací.

Příklad: Zobrazení rozdělení dat pomocí shlukovací metody do grafu.

Vstup: Matice z obrázku 2.

Výstup: Grafické zobrazení výsledných shluků.



Obrázek 3: Příklad - grafické zobrazení shluků

Na obrázku 3 vidíme příklad rozdělení dat do jednotlivých shluků.

2.2 Použití data miningu

S narůstajícím množstvím uchovávaných dat v souborech, databázích a dalších repozitářích, je často důležité tyto data později analyzovat. Cílem takové analýzy je vytěžit z dat prospěšné informace, které nejsou na první pohled viditelné.

2.2.1 Co můžeme v datech nalézt

To, jaká data a vztahy mezi nimi můžeme nalézt, závisí částečně od toho, co hledáme a co považujeme za relevantní výstup. Rozlišujeme dva typy data mingových úloh, které si můžeme popsat.

- Deskriptivní data mining: Popisují hlavní vlastnosti existujících dat. Např.: šetření odchylek v datech, shlukování nebo asociace.
- Prediktivní data mining: Předpovídají chování na základě dostupných dat z minulosti. Např.: klasifikace a regresní analýza.

Obecně lze v datech nalézt:

- **Charakteristická data:** Výsledné zhodnocení hlavních atributů objektů.
- **Asociace (vztahy):** Asociace neboli asociační pravidla, nám určují, jak jedna množina atributů vyplývá z druhé množiny atributů.

- **Klasifikace:** Data rozdělená do konečného počtu skupin mají nějakou společnou vlastnost. Poté hledáme a klasifikujeme nové objekty na základě těch předešlých.
- **Shluky:** Podobně jako klasifikace, rozdělují shluky data do skupin. Na rozdíl od klasifikace ve shlucích nemáme informaci o společném jmenovateli ve skupině. Shlukovaní se snaží rozdělit tyto data do skupin, za pomocí hledání podobností nebo vzdáleností mezi objekty v jedné skupině.

2.3 Oblasti využití data miningu

Oblasti využití data miningových metod v praxi jsou poměrně široké. Neklade se zde důraz na striktní rozdělení do určitých segmentů. Primárním požadavkem k rozpracování analýzy, je dispozice využitelných dat u institucí nebo soukromých osob. Cílenou aplikací jedné z data miningových metod mohou jednotlivé subjekty například vylepšit řadu svých služeb. Ačkoliv tyto metody mají rozsáhlé pole působnosti, uvedu zde několik odvětví, ve kterých se s nimi můžeme setkat nejčastěji.

2.3.1 Marketing

Jedna z oblastní, která hojně využívá výše definovaný sběr dat, je oblast marketingu. Své metody a techniky aplikuje výhradně pro seznámení se s potřebami zákazníků. Díky analýze spotřebitelského chování, je možné zdokonalit celkovou marketingovou strategii podniku, a tím umocnit i efektivitu reklamních kampaní [5].

Tuto analýzu lze provádět jak v kamenných prodejnách, tak v internetových obchodech. Zcela výhodnější postavení mají e-shopy. Díky jejich digitální povaze, mohou jejich majitelé získat dodatečné informace z jejich soukromých databází. Nabízí se možnost sledování statistik prohlížených produktů nebo selekce nejoblíbenějších kategorií zboží.

Základním předpokladem ke zdokonalení služeb je potom schopnost umět získané informace využít. V kamenných prodejnách je analýza využívaná k umístění výrobků na prodejně. Internetové obchody mohou ke zvolenému výrobku předložit uživatelům související zboží, které si jiní zákazníci také oblíbili. Zvýšit obraty internetových obchodů. Díky analýze nákupního košíku, dokážeme určit pravděpodobnost nákupu jednoho zboží spolu s jiným (např. chléb => máslo).

2.3.2 Educational Data Mining

Pokud použijeme data miningu v e-learningových systémech, jedná se o speciální odvětví zvané Educational Data Mining (EDM). Obecně se v těchto systémech nachází velké množství užitečných dat. Data jako jsou například (kolikrát student absolvoval určitý test, jaký byl jeho celkový čas, jaká otázka mu dělala největší problém). Za pomocí shlukové analýzy pak dokážeme tyto studenty rozdělovat do jednotlivých skupin podle chování. EDM se zabývá autor publikace [6].

Dále můžeme analyzovat:

- **Předpověď úspěšnosti studenta** - za pomoci nasbíraných dat v minulosti. Můžeme předpovědět, že pokud se student chová podobně, budeme očekávat i podobný výsledek.
- **Odhalení obtížnosti studijního materiálu** - dokážeme porovnat procentuální úspěšnosti v jednotlivých testech. Ty nám mohou odhalit, zda je test nekorektní, příliš složitý nebo špatně formulovaný.

EDM je z hlediska data miningu složitější než marketing. Nachází se zde velké množství kategoriálních (nenumerických) dat, které je potřeba upravit pro jednotlivé metody analýzy.

2.3.3 Jiné

Například můžeme využít data mining v medicíně nebo telekomunikacích. Můžeme sledovat efektivitu léčby, porovnáváním symptomů a podávaných léků [7]. Predikovat neplatiče telefonních služeb [8].

V praxi můžeme data mining a jeho metodiky používat v jakémkoliv odvětví, které má dostatečně velké množství dat ke zpracování.

3 Metody použité pro analýzu dat

V následující kapitole si popíšeme vybrané metody, které byly použity při analýze dat e-learningového systému eLogika. Statistické metody, shluková analýza a asociační pravidla, tvoří tři základní stavební kameny pro analýzu chování uživatelů v systému. Popíšeme si detailněji jednotlivé metody a společně s příklady se jim pokusíme snadněji porozumět.

3.1 Statistické metody

V literatuře bývají označovány za základní analýzu, kterou můžeme na data aplikovat. Část autorů považuje statistické metody za součást analýzy dat a do data miningových metod je nezahrnují. My se ovšem zaměříme pouze na nejzákladnější statistické metody. Přehledy všech metod máme k dispozici v přiložené literatuře [9].

3.1.1 Průměr

Ve statistice nejčastěji pracujeme s klasickým aritmetickým průměrem pozorovaných hodnot. Existují i další míry, jako například průměr geometrický, kvadratický nebo harmonický. Pro naše účely bude úplně dostačující již zmíněný aritmetický průměr značen $\bar{x}$.

S průměrem se v běžném životě setkáváme velice často. Ne vždy poskytuje správné informace o datech. U průměrné hodnoty můžeme totiž narazit na problém, odlehlých pozorování. K těmto zmíněným problémům dochází v případech, kdy data obsahují extrémní hodnoty.

Vzorec pro výpočet aritmetického průměru:

$$\overline{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

Příklad: Mějme soubor prvků X = [10, 20, 30, 40, 555].

Lze na první pohled vidět, že číslo 555 je extrémní hodnota. Průměr tohoto souboru je 131 a ve výsledku čtyři z pěti hodnot tohoto průměru ani zdaleka nedosahují. Pokud odstraníme odlehlé pozorovaní, výsledný průměr souboru bude číslo 25, které nám charakterizuje daný soubor.
Vyrušíme-li odlehlá pozorování, zvýšíme celkovou validitu průměrné hodnoty prvků v souboru.

3.1.2 Medián

Medián, na rozdíl od průměru, je méně náchylný k výskytu extrémních hodnot daného souboru. Další jeho výhodou, je schopnost aplikace pro jakékoliv hodnoty, které se dají nějakým způsobem seřadit.

Vzorec pro výpočet mediánu:

$$Median(X) = \frac{x_{N/2} + x_{(N/2)+1}}{2}$$

Příklad: Máme soubor X = [1,1,2,3,5,8,13].

Dosadíme-li jej do vzorce, výsledek bude Median(X) = 3.

3.1.3 Modus

Modus nám určuje nejčastěji se vyskytující hodnotu ve statistickém souboru. Dá se použít na libovolný soubor prvků, které se nemusejí seřazovat (např. koza, petržel, jablko). Je označován $\tilde{x}$.

Příklad: Mějme soubor M = [X,X,Y,Y,Z].

Protože v souboru se může nacházet více hodnot s nejvyšší četností. Výsledný modus může nabývat vice hodnot. V tomto případě je výsledný modus X a Y.

3.1.4 Rozptyl

Rozptylem se udává, jak moc jsou hodnoty ve statistickém souboru rozptýleny. Rozptýlením se rozumí, jak moc se hodnoty vzdalují od průměrné hodnoty. Velká hodnota rozptylu nám naznačuje, že se některé hodnoty ve statistickém souboru výrazně odchylují od hodnoty průměrné. Rozptyl značíme jako o^2 a je to tedy druhá odmocnina směrodatné odchylky.

Vzorec pro výpočet rozptylu:

$$Var(X) = \frac{1}{N} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2$$

Příklad: Máme dva soubory se stejným aritmetickým průměrem, ale odlišným rozptylem X = [5,7,9,11,16,18] a Y = [1,2,4,5,7,47].

Průměr souboru X a Y je 11. Rozptyl u souboru X je 21,7 a u množiny Y je 263. Zde můžeme pozorovat, že soubor Y disponuje příliš velkým rozptýlením jednotlivých hodnot. Proto můžeme předpokládat, že obsahuje extrémní hodnoty. V tomto příkladě (číslo 47).

3.1.5 Směrodatná odchylka

Značíme σ, jelikož rozptyl získáme jako druhou mocninu směrodatné odchylky, tak analogicky směrodatná rozptylka se počítá jako odmocnina z rozptylu.

Vzorec pro výpočet směrodatné odchylky:

$$\sigma = \sqrt{Var(X)} = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2}$$

Příklad: Mějme soubor z předešlého příkladu X=[5,7,9,11,16,18], kde rozptyl je 21,7.

Dosadíme tedy do vzorce pro výpočet směrodatné odchylky:

$$\sigma(X) = \sqrt{Var(X)} = \sqrt{21,7} = 4,65$$

3.1.6 Porovnání jednotlivých metod

Pro porovnání si zavedeme do tabulky tři soubory a jejich výsledné hodnoty.

X = [9,13,15,15,16,23,28,29,32,42,53]
Y = [5,6,7,8,14,31,31,33,33,43,64]
Z = [12,13,15,15,16,23,28,29,32,135,93]

Soubor	Počet hodnot	Průměr	Medián	Modus	Rozptyl	Směr. odchylka
X	**11**	**25**	**23**	**15**	166,55	12,91
Y	**11**	**25**	31	31,33	321,8	17,94
Z	**11**	37,36	**23**	**15**	1428,6	37,8

Tabulka 7: Porovnání statistických metod

Soubory X a Y mají stejný aritmetický průměr, ale značně rozdílný rozptyl. Soubory X a Z mají stejný medián, modus a také značně rozdílný rozptyl. Můžeme tedy vypozorovat, že určení pouze průměru v souboru hodnot není dostačující. Ostatní metody statistické analýzy nám pomáhají detailněji porozumět zkoumaným datům.

3.2 Shluková analýza

Shlukování je rozdělení dat do skupin podle vzájemné podobnosti. Cílem této analýzy je nalezení ideálního seskupení dat. Takové seskupení vykazuje minimální rozptyl hodnot uvnitř shluku za současného maximálního rozptylu mezi těmito shluky.

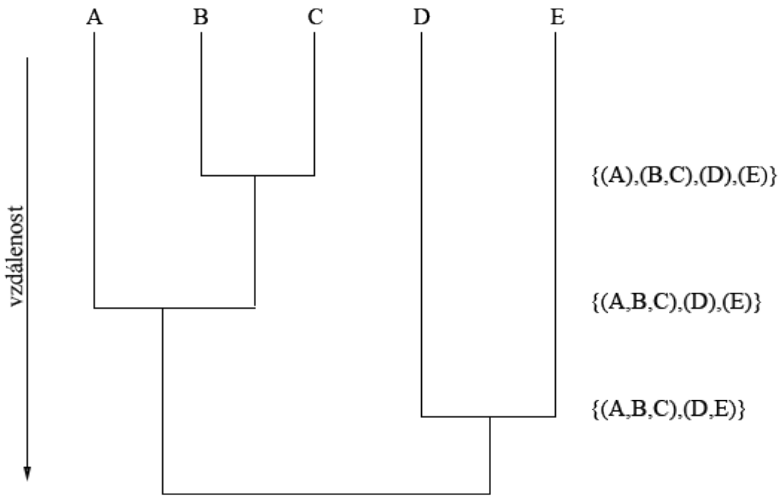
Shluková analýza se zabývá metodami a algoritmy, pomocí kterých dokážeme rozdělit data s podobnými vlastnostmi do stejných skupin. Nejčastěji se setkáme s použitím shlukové analýzy spolu s dalšími metodami. Díky nim jsme schopni maximálně využít potenciál zkoumaných dat. Jakmile rozdělíme prvky do jednotlivých shluků, můžeme identifikovat jejich charakteristické vlastnosti.

Pomocí shlukování tak dokážeme najít vztahy mezi jednotlivými objekty, které nám poskytují důležité informace o struktuře dat. Shlukové analýze se věnuje autor publikace[10].

V této podkapitole si tedy popíšeme základní rozdělení shlukování. Dále se budeme zabývat standardizací dat, redukcí dimenze, metodami pro určení optimálního počtu shluků a způsoby měření podobnosti objektů.

3.2.1 Hierarchické shlukování

Tato metoda patří společně s metodou rozkladu (partitioning) mezi nejzákladnější metody shlukování dat. U hierarchického shlukování začínáme vždy s N-triviálními shluky, kde N vyjadřuje počet prvků množiny. Postupně generujeme N-1 shluků, dokud nám na konci výčtu nezůstane jeden shluk se všemi pozorovanými daty. V každém kroku tak shlukujeme do sebe dva nejbližší shluky. Pomocí dendogramu (viz obr. 4), můžeme znázornit jednotlivé rozklady při hierarchickém shlukování.



Obrázek 4: Dendogram hierarchického shlukování

3.2.2 Nehierarchické shlukování

Jak už z názvu vyplývá, tyto metody shlukování nevytvářejí hierarchickou strukturu. Rozdělují jednotlivé objekty do předem určeného počtu shluků. Zadaný počet shluků se dále nerozděluje. V několika opakovaných průchodech se snaží dosáhnout nejlepšího rozdělení objektů pro daný počet shluků.

- **MacQueenova metoda**
  MacQueenova metoda neboli metoda K-means je iterační algoritmus, minimalizující součet vzdálenosti každého objektu od těžiště (centroidu) shluku. Cílem této metody je získání k shluků, které jsou kompaktní a současně navzájem dobře oddělené.

  Výhoda: Jedná se o jednoduchý algoritmus se snadnou implementací. Algoritmus na vstupu přijímá fixní počet k shluků. Ten dále zajistí rozřazení jednotlivých dat do individuálních shluků tak, aby byly shluky co nejvíce rozdílné a zároveň redukoval jejich vnitroskupinovou variabilitu.

  Nevýhoda: Citlivý na odlehlá pozorování.

3.2.3 Standardizace dat

Před tím, než začneme používat některou ze shlukovacích metod, je doporučováno předem zanalyzovat data, se kterými budeme pracovat. Respektive data, která poskytneme funkci ke zpracování analýzy.

Měli bychom zvážit, zda použít standardizaci dat, nebo data nechat v původním formátu. Pokud jsou data ve stejných jednotkách (metry, sekundy, kilogramy) a se stejnou variabilitou (malým rozptylem, směrodatnou rozptylkou), není nutné provádět standardizaci.

V případě, že máme vysokou variabilitu dat, mají v procesu velký vliv při určování podobnosti a nepodobnosti dat. Za takových podmínek doporučuji dané data, před aplikováním shlukovacích metod, standardizovat. Více o standardizaci dat a některých metodách v literatuře [11].

3.2.4 Redukce dimenze

Různé problémy s velikostí vícerozměrných dat, můžeme vyřešit pomocí redukce dimenze. Typicky se můžeme setkat s redukcí vícerozměrných dat do dvou nebo tří dimenzí, které se dají jednoduše zobrazit do grafu.

Může posloužit jako nástroj pro zobrazení jednotlivých shlukovaných objektů ve 2D nebo 3D prostoru, což lze snadno vizualizovat.

- **Metoda hlavních komponent**
  Metoda hlavních komponent (Principal Component Analysis neboli PCA) nám umožnuje redukovat počet původních proměnných na co nejmenší počet hlavních komponent, které takto popisují vícerozměrné rozdělení hodnot. To vše bez velké ztráty datové informace.

  Jelikož se stává, že velké množství zkoumaných dat je pro interpretaci nepřehledné, je nutno zjistit, zda by nešlo reprezentovat tyto objekty menším počtem, třeba i uměle vytvořených proměnných. Jak redukovat dimenze pomocí PCA je detailně popsáno v článku [12].

3.2.5 Nalezení optimálního počtu shluků

Před aplikováním shlukovacích metod je důležité si uvědomit, na kolik shluků data rozdělit. Touto tématikou se zabývá například autor publikace [13]. Popravdě, neexistuje doposud žádný způsob, jak určit správný počet shluků. Dokážeme tedy najít pouze optimální počet shluků, podle určitých kritérii. Existuje několik metod, které se zabývají touto problematikou.

- **Silhoulette index**
  Silhoulette index může nabývat hodnot s intervalu od -1 do 1. Větší číslo znamená, že objekt je správně zařazený ve vlastním shluku a nepatří do ostatních shluků. Algoritmus pracuje do doby, než najde většinu objektů s vysokou hodnotou silhoulette indexu.

  Pokud ale převažují hodnoty nízké nebo záporné, počet shluků je buďto příliš nízký nebo příliš vysoký.

  Silhoulette index můžeme používat s jakoukoliv metodou pro měření vzdálenosti např. Euklidovskou vzdálenost nebo Manhattan viz podkapitola 3.2.6

  Vzorec:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

  $a(i)$ – průměrná hodnota (ne)podobnosti objektu i s ostatními objekty ve stejném shluku určuje, jak dobře je shluk sestaven.
  $b(i)$ –nejmenší průměrná hodnota (ne)podobnosti objektu i, k ostatním shlukům, neobsahující tento prvek určuje nejbližší shluk, ke kterému by zmíněný prvek mohl být přiřazen.

- **K-fold Cross-validation**
  Metoda využívá kombinací více testů pro získání odhadu modelové chyby. Data jsou náhodně rozdělena do k-skupin o stejné velikosti. Jedna část z k subsetů je použitá pro testování a druhá část k-1 subsetů je použitá jako tzn. training set. Tento proces se opakuje pro všechny data. Výsledkem jsou jednotlivé chybovosti při zvoleném k. V praxi se většinou využívá 10-fold cross validation, kde k=10.

3.2.6 Měření vzdálenosti

Za pomocí měření vzdáleností dvou objektů, dokážeme určit jejich rozdílnost. Čím více jsou od sebe dva objekty vzdáleny, tím více jsou si odlišné.

Vzdálenost může nabývat hodnot v intervalu od -1 do 1. Nultá hodnota znamená totožnost objektů.

Pro měření vzdálenosti mezi dvěma vektory existuje celá řada metod. Více o jednotlivých metodách se můžeme dozvědět od autorů publikací [10] a [14].

Mezi nejznámější metody měření vzdáleností patří Euklidovská a Manhattanská. Obě si níže detailněji popíšeme.

- **Minkowského vzdálenost**
  Obecná metoda pro měření vzdáleností. Zaleží na parametru λ.

  Vzorec:
  $$d_{ij} = \sqrt[\lambda]{\sum_{k=1}^{n}(x_{ik} - x_{jk})^{\lambda}}$$

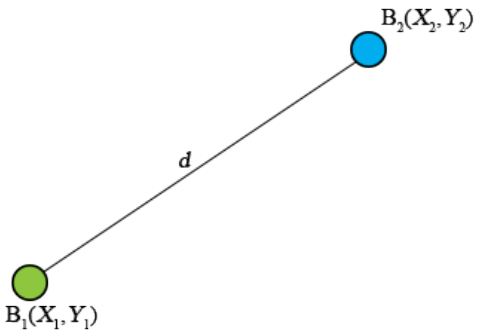
- **Euklidovská vzdálenost**
  Euklidovská vzdálenost je nejznámější a nejvíce používaná metoda pro měření vzdálenosti. Vzdálenost mezi dvěma vektory X a Y je založena na Pythagorově větě. Euklidovská vzdálenost je speciální případ minkowského vzdálenost s $\lambda = 2$. Pokud chceme měřit vzdálenosti kategoriálních dat, tak se převádí na tzv. „Dummy proměnné“, které nabývají hodnot 0 nebo 1.

  Vzorec:
  $$d_{ij} = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^{2}}$$

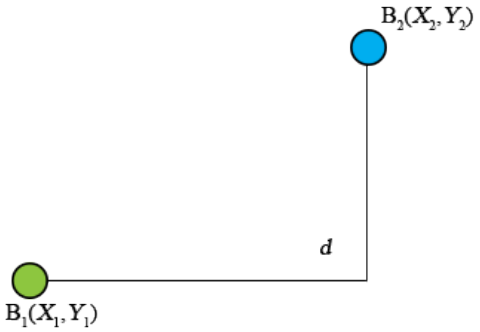
- **Manhattanská vzdálenost**
  Součtem vzdáleností všech souřadnic mezi vektory vypočteme manhattanskou vzdálenost. Neměříme tedy vzdálenost diagonálně jako v případě euklidovské vzdálenosti.

  Vzorec:
  $$d_{ij} = \sum_{k=1}^{n}|x_{ik} - x_{jk}|$$

3.2.7 Porovnání Euklidovské a Manhattanské vzdálenosti



Obrázek 5: Euklidovská vzdálenost



Obrázek 6: Manhattanská vzdálenost

3.3 Asociační pravidla

Asociační pravidla se obecně používají v transakčních databázích, kde jedna transakce je tvořena množinou prvků. Procesem analýzy asociačními pravidly získáváme zajímavé vztahy mezi velkým množstvím dat v databázích.

Jejich hlavním úkolem je nalezení vztahů (asociací) mezi položkami. Nejčastějším příkladem, který se používá k porozumění asociačních pravidlech je analýza nákupního košíku (market based analysis). Asociační pravidla a analýzu nákupního košíku podrobněji vysvětlují autoři publikace [15].

3.3.1 Podpora a spolehlivost

Každé z jednotlivých pravidel mají samozřejmě jinou vypovídající hodnotu. Existují pravidla, která jsou více spolehlivá a naopak. Proto při generovaní těchto pravidel používáme dvě metriky, které jsou základní mírou relevance asociačních pravidel.

- **Podpora (Support)**
  Uvádí pravděpodobnost výskytu všech predikátů $i_1, i_2, \ldots, i_n$ v datech.

  Vzorec:

$$Podpora(T) = \frac{počet\ transakcí\ obsahujících(X\&Y)}{celkový\ počet\ všech\ transakcí} * 100$$

- **Spolehlivost (Confidence)**
  Určuje podmíněnou pravděpodobnost jevu i_n v datech s podmínkou, že platí i_1 až i_n.

  Vzorec:

$$Spolehlivost(T) = \frac{počet\ transakcí\ obsahujících(X\&Y)}{počet\ transakcí\ obsahujících(X)} * 100$$

3.3.2 Příklad analýzy nákupního košíku

Zkoumaná data z analýzy nákupního košíku prochází databáze, která obsahuje všechny transakce. V tomto případě je transakcí obsah nákupního košíku. Analyzovaná data jsou v booleovské formě (mohou nabývat pouze hodnoty 0 nebo 1). Ve smyslu např.: 1 – nakoupil, 0 - nenakoupil (viz tab. 9).

Obsahuje:

- Položky (Items) – jednotlivé proměnné (produkty)
- Transakce (Transactions) – jednotlivé transakce obsahující jednu nebo více položek
- Tabulka četností – matice četností libovolných dvou položek
- Tabulka výsledných asociačních pravidel

id položky	Název
1	Maso
2	Rýže
3	Máslo
4	Chléb
5	Sůl

Tabulka 8: Příklad asociační pravidla - tabulka jednotlivých položek

id transakce	Položka 1	Položka 2	Položka 3	Položka 4	Položka 5
1	1	1			1
2			1	1	1
3		1			
4	1	1	1		
5		1			1
6	1	1			
7	1		1		1
8		1		1	1
9	1	1			1

Tabulka 9: Příklad asociační pravidla - tabulka transakcí

	Položka 1	Položka 2	Položka 3	Položka 4	Položka 5
Položka 1	5	4	2	0	3
Položka 2	4	7	1	1	4
Položka 3	2	1	3	1	2
Položka 4	0	1	1	2	2
Položka 5	3	4	2	2	6

Tabulka 10: Příklad asociační pravidla - tabulka četností jednotlivých položek

Minimální podpora, p = 20%
Minimální spolehlivost, s = 60%

Z výsledků (viz tab. 11), které byly podrobeny asociačními pravidly si můžeme všimnout, že kdykoliv zákazník koupí chléb, zároveň k němu dokoupí i sůl. Tento předpoklad má stoprocentní spolehlivost. Pro kontrolu můžeme nahlédnout do tabulky transakcí a porovnat výsledek s tvrzením. Pokud je chléb (položka 1) s hodnotou 1 (koupeno), vždy je v transakci i sůl (položka 2)

s hodnotou 1.

Výsledkem této analýzy jsou asociační pravidla ve tvaru:

$$a_1 \wedge a_2 \wedge a_{n-1} \Rightarrow a_n$$

Levou stranou pravidla se rozumí předpoklad (antecedent) a pravá strana znamená závěr (sukcedent). Symbolům a_n se v asociačním pravidle říká predikáty stejně jako v matematické logice.

Čtěme výraz: Jestliže si zákazník koupil $a_1, a_2, \ldots a_{n-1}$, pak si koupí i zboží a_n se spolehlivostí a podporou p.

Předpoklad (antecedent)	**Závěr (sukcedent)**	**Spolehlivost (confidence)**
Chléb	Sůl	100%
Maso	Rýže	80%
Máslo	Maso	66,7%
Sůl	Rýže	66,7%
Máslo	Sůl	66,7%
Maso & Sůl	Rýže	66,7%
Maso	Sůl	60%

Tabulka 11: Příklad asociační pravidla - výsledná asociační pravidla

Užití výsledku analýzy:

Obecně tato analýza může sloužit pro strategické umístění zboží, které se nejčastěji kupuje společně. Paradoxně jsou tyto komplementy umisťovány na prodejně co nejdále od sebe. Tato marketingová taktika zvyšuje šance při doplňkovém prodeji. Zákazník je nucen projít celou prodejnou a čelit tak nástrahám v podobě zboží, které v danou chvíli není pro něj prioritní.

4 Systém eLogika

eLogika je LMS e-learningový systém, který byl primárně navržen pro předmět Matematická logika. Hlavním důvodem vzniku tohoto systému je usnadnění výuky studentům i vyučujícím. Studenti mají možnost nahlížet do studijních materiálů vždy a všude díky přítomnosti všech informací na internetu. Pro vyučující tento systém disponuje funkcemi jako je například automatické generování a vyhodnocování testů, které šetří čas [16].

V této kapitole se budeme věnovat analýze poskytnutých dat v systému eLogika. Podrobněji si rozepíšeme, jak vypadají logy přístupů a navrhneme si možná vylepšení.

4.1 Analýza poskytnutých dat

Dříve, než se dostaneme k návrhu a implementaci aplikace si popíšeme, jaká data máme v systému k dispozici a jak se k nim dostaneme.

Systém eLogika komunikuje s aplikacemi třetích stran prostřednictvím REST, kde komunikace mezi klientem a serverem probíhá přes HTTP protokol.

V této podkapitole si objasníme, co je to REST a jaké dostupné metody budeme používat k analýze dat.

4.1.1 Representational State Transfer

V roce 2000 spoluzakladatel protokolu HTTP, Roy Fielding uvedl ve své dizertační práci Architectural Styles and the Design of Network-based Software Architectures [17] návrh architektury REST.

Pomocí REST můžeme jednoduše vytvářet, číst, mazat a editovat data na serveru pomocí http dotazů.

V REST jsou definovány čtyři základní přístupy ke zdrojům:

- **GET**
  HTTP GET se používá pro čtení dat ze zdroje. Můžeme dostat tyto HTTP odpovědi na náš dotaz kód 200 (OK). Pokud nastane někde chyba, můžeme dostat odpovědi typu 404 (nenalezeno) nebo 400(špatný dotaz).

  Příklad:F

  - GET `http://mojepage.cz/zobraz/123`
  - GET `http://mojepage.cz/zobraz.aspx?user=5`

  Výstup bývá většinou ve formátu XML nebo JSON.

- **POST**
  Používá se pro vytvoření nebo odesílaní dat na server. Pokud se data úspěšně podaří odeslat, dostaneme od serveru odpověď HTTP 201.

  Příklad:

  – POST `http://mojepage.cz/pridej`

- **PUT**
  Používá se pro vytvoření nebo odesílaní dat na server. Zároveň pokud data již existují, aktualizuje je.

  Příklad:

  – PUT `http://mojepage.cz/edituj`

- **DELETE**
  U DELETE metody dochází ke smazání zadaných dat ze serveru. Můžeme dostat tyto odpovědi ze serveru: HTTP 204 (NO CONTENT), HTTP 200 (OK).

  Příklad:

  – DELETE `http://mojepage.cz/vymaz/123`

4.1.2 Popis použitých metod

V této sekci jsou popsány všechny metody, které budeme potřebovat pro analýzy. Ke každé z metod je jednotlivě uvedeno, jak se dá zavolat a jaké můžeme očekávat výstupy.

- **Seznam škol**

  GET *api/School/allschools*

  Metoda vrací seznam všech škol v systému eLogika.

- **Seznam kurzu ve škole**

  GET *api/Course/allcourses?schoolId={0}*

  Metoda vrací seznam všech kurzů pro jednotlivé školy.

  Vstupní parametry URL:
  schoolId - identifikátor školy, pro kterou chceme zobrazit jednotlivé kurzy

- **Seznam školních roků pro kurz**

  GET *api/Course/allyearsbycourse?courseId={0}*

  Metoda vrací seznam všech školních roků, ve kterých probíhal daný kurz.

  Vstupní parametry URL:
  courseId - id kurzu, pro který chceme zobrazit jednotlivé školní roky

- **Seznam studentů v kurzu**

  GET *api/StudentsSummary/studentsbycourseinfo?courseInfoId={0}&formaStudia={1}*

  Metoda vrací seznam všech studentů, kteří jsou/byli přihlášeni v daném kurzu.

  Vstupní parametry URL:
  courseInfoId - id kurzu
  formaStudia - forma studia (prezenční, kombinované)

- **Seznam hodnocení uživatele**

  GET *api/StudentEvaluation/getdata?courseInfoId={0}&userId={1}*

  Metoda vrací hodnocení pro jednotlivé uživatele.

  Vstupní parametry URL:
  courseInfoId - id kurzu
  userId - id uživatele

- **Logy aktivit v systému**

  GET *api/User/getlogaccessbyiduzivatel?idUzivatel={0}&casOD={1}&casDO={2}*

  Metoda vrací aktivity v systému konkrétního uživatele v zadaném časovém intervalu.

  Vstupní parametry URL:
  idUzivatel - id uživatele
  casOD - datum a čas od kdy chceme zobrazit logy
  casDO - datum a čas do kdy chceme zobrazit logy

- **Informace o semestru**

  GET *api/Semester/semestersbycourseinfo?courseInfoId={0}*

  Metoda vrací informace o začátku a konci semestru atd.

  Vstupní parametry URL:
  courseInfoId - id kurzu

- **Získaní id uživatele podle loginu v systému**

  GET *api/User/GetUserIdBySchoolsLogin?schoolId={0}&schoolLogin={1}*

  Metoda vrací id v databázi podle školního loginu.

  Vstupní parametry URL:
  schoolId - id školy
  schoolLogin - školní login

- **Role v systému**

  GET *api/Role/getroles?userId={0}&schoolId={1}*

  Metoda vrací seznam rolí, které má uživatel v dané škole.

  Vstupní parametry URL:
  userId - id uživatele
  schoolId - id školy

- **Seznam aktivit v kurzu**

  GET *api/CourseConditions/groupactivites?courseInfoId={0}&userId={1}&roleId={2}*

  Metoda vrací seznam všech aktivit uživatele v kurzu.

  Vstupní parametry URL:
  courseInfoId - id kurzu
  userId - id uživatele
  roleId - id role

- **Seznam školních tříd**

  GET *api/SchoolClass/getschoolclassesbycourseinfo?courseInfoId={0}*

  Metoda vrací seznam školních tříd v kurzu.

  Vstupní parametry URL:
  courseInfoId - id kurzu

- **Seznam studentů ve třídě**

  GET *api/User/GetStudentOfTrida?classId={0}*

  Metoda vrací seznam studentů v zadané třídě.

  Vstupní parametry URL:
  classId - id třídy

- **Login do systému**

  POST *api/User/loginsecureobject*

  Metoda pro přihlášení do systému.

  Vstupní parametry:
  Login - login uživatele
  Password - heslo uživatele

4.2 Analýza logovaných dat

Logovaná data jsou souhrnné seznamy akcí, které jsou vykonány na straně klienta. Tyto logy zůstávají na straně serveru. Logovaná data, která webový server shromažďuje, obsahují informace o uživatelích, kteří přistupují k webovému obsahu v systému. Velikost logu vzrůstá rapidně kvůli četnosti uchovávaní dat. Analýza těchto robustních dat vyžaduje využití data miningových metod (viz kapitola 3).

V této podkapitole se budeme detailněji věnovat REST metodě getlogaccessbyiduzivatel (viz kapitola 4.1.2)

4.2.1 Struktura logovaných dat přístupů v systému eLogika

V této podkapitole si ukážeme, jakým způsobem jsou logované data přístupů v databází a REST API.

(a) **Struktura tabulky přístupů v databázi**
Webové logy přístupů jsou zpracovávány v každé webové aplikaci rozdílně. Z těchto důvodů je potřeba provést analýzu logovaných dat v systému eLogika. Základní informace zpracovávané v systému eLogika jsou:

- *id_log*: unikátní identifikátor logu
- *id_uzivatel:* : identifikátor uživatele, u kterého byl přístup zaznamenán
- *id_role*: v systému je několik rolí (student, tutor, garant), tento sloupec v tabulce určuje, pod jakou rolí byl uživatel přihlášen, když vykonával tuto aktivitu
- *id_skola_info*: tak jako je několik rolí, může být uživatel v několika školách, proto identifikátor školy, ke které je uživatel přihlášen
- *id_info_kurz*: identifikátor kurzu, pod kterým je uživatel přihlášen
- *cas*: datum a čas generovaní dotazu na server
- *url*: zdroje, ke kterým přistupuje uživatel, v našem případě stránky s příponou „.aspx“
- *ip*: detekce IP adresy stroje přistupujícího ke zdroji

Toto je obsah každého logu v systému. Tyto logy jsou primárně zaznamenávány pro statistiku používání webu a zjišťování chování uživatelů v systému a samozřejmě následovného zlepšovaní struktury webu.

id_log	id_uzivatel	id_role	id_skola_info	id_info_kurz	cas	urp	ip
1	2	5	12	80	2013-09-10 23:31:01.927	/Pages/Default.aspx	192.168.1.1
2	2	5	12	80	2013-09-10 23:31:05.803	/Pages/Student/Rozvrh.aspx	192.168.1.1
3	5	3	11	83	2013-09-10 23:33:06.853	/Pages/Default.aspx	192.172.1.5
...	...	...	...	...	...	...	...
N	12	5	10	92	2015-01-12 12:10:01.153	/Pages/VyberSkoly/VyberSkoly.aspx	191.112.1.1

Tabulka 12: Příklad logů

(b) **Struktura dat přístupů přes REST API**
Abychom dostali tyto informace do naší aplikace, máme k dispozici metodu getlogaccessbyiduzivatel (viz kapitola 4.1.2). Vstupem k této metodě je *user_id* a *course_id* a časový interval od - do. Nedostaneme tedy celou databázi všech logů, ale musíme znát konkrétní *id_uzivatele* a kurz, ve kterém studuje, abychom zobrazili jeho logy přístupů.
Tato metoda obsahuje stejné informace, jako tabulka viz 13.

Popis atributů:

- *UserId*: identifikátor uživatele, u kterého byl přístup zaznamenán
- *RoleId* : identifikátor role v systému (student, tutor, garant)
- *SkolaInfoId* : identifikátor školy, ve které je uživatel přihlášen
- *KurzInfoId* : identifikátor kurzu, ve kterém je uživatel přihlášen
- *Cas* : datum a čas přístupu k dané stránce
- *URL* : adresa logované stránky
- *IP* : IP adresa zdrojového počítače
- *OS* : Operační systém zdrojového počítače

Přes REST API dostaneme stejné informace, jaké jsou obsaženy v databázi. Jediná změna se týká pouze názvů jednotlivých atributů, kdy v databázi máme například sloupec *id_uzivatel*, který je pod názvem *UserId*.

4.2.2 Úprava logovaných dat

Data obsažená v logu nemůžeme, bez provedení úprav, používat pro miningové procesy. Proto obsah těchto logů musí být zpracován [18]. Data, která jsou nepotřebná, jsou odstraněna neboli minimalizovaná pro následovné použití.

- **Čištění dat**
  Data přístupů, které systém eLogika zaznamenává, nemusí být vždy ve správném tvaru pro data miningové metody. Uvedeme si tedy příklady, o jaká data se jedná a jak je upravíme.

  1. *Neplatná URL stránka*
     Ve výpisu jednotlivých logů jsem narazil na případy, kdy se místo URL adresy v logu nachází text (např. Error nebo Unknown). Tyto logy je potřeba filtrovat.
  2. *Parametr v URL adrese*
     Dále se v systému objevují i logy s GET parametry (např. `/Pages/Other/Termin.aspx?ID_SATA=A_60`).
     Ty stačí pouze upravit a to odstraněním textu s otazníkem `/Pages/Other/Termin.aspx?ID_SATA=A_60` na `/Pages/Other/Termin.aspx`.

3. *Názvy URL adres*
   V ojedinělých případech jsem narazil na odlišné názvy jednotlivých stránek. Například `/Pages/Student/Rozvrh.aspx` a `/Pages/Student/rozvrh.aspx`.
   Řešením může být převedení všech URL na malá písmena.
4. *Neplatný datum a čas*
   Pokud v logu narazíme na neplatný datum a čas, tato informace nemá pro nás žádnou vypovídající hodnotu a musíme tyto data zahodit.

- **Převod dat**
  Abychom mohli analyzovat odlišnost uživatelů z hlediska chování v systému, budeme potřebovat četnosti jednotlivých aktivit. Tyto poté převedeme na jednotlivé vektory, které se pak dají lehce předat miningovým metodám. Převod dat můžeme nastínit na příkladu.

Příklad: V systému máme statisíce logů přístupů, které jsou zaznamenávány postupně podle času, ve kterém nastaly. Pro naši analýzu potřebujeme chování jednotlivých uživatelů v systému, ne chování všech uživatelů najednou (takové chování by se nedalo ani s ničím dále porovnávat).

Vybereme si z logů jednotlivé uživatele a u nich spočítáme četnosti návštěv jednotlivých stránek (aktivit).

Uživatel/Aktivita	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**
Uživatel1	50	41	44	32	12	10	5	4	1
Uživatel2	39	38	33	28	12	12	8	3	0
Uživatel3	25	22	21	20	14	14	7	7	0
Uživatel4	34	31	30	19	18	22	12	4	1
Uživatel5	52	48	29	16	11	7	6	2	1

Tabulka 13: Příklad četnosti logů

Čteme: *„Uživatel1 navštívil stránku A1 celkem 50 krát.“* nebo *„Uživatel1 vykonal aktivitu A1 celkem 50 krát.“*

Z takto vytvořené tabulky můžeme jednoduše získat vektor jednotlivých studentů, který popisuje jejich chování v systému.

Příklad: *Uživatel1* můžeme převést na vektor o 9-ti dimenzí (50,41,44,32,12,10,5,4,1).

Tohle je jen základní příklad převodu dat z logů na vektor popisující chování. Samozřejmě můžeme do výsledné tabulky přidat další hodnoty, které více popisují chování uživatelů (četnosti rozdělené do kategorii ráno, odpoledne, večer), průměrná doba strávená na stránce atd.

4.3 Návrh dodatečných dat k logování

Součástí zadaní diplomové práce je i návrh dodatečných dat, které se mají v eLogice logovat. Navržení dodatečných dat má pomoci k zefektivnění vypovídajících informací aplikace. Uvedu dva návrhy, které byly schváleny vedoucím diplomové práce.

- **Logování dat z mobilní aplikace**
  V rámci diplomové práce [19] byla vytvořena mobilní aplikace systému eLogika. Tuto platformu bude v budoucnu preferovat více studentů a dá jí přednost před klasickou webovou aplikací. Doporučil bych logování přístupů jednotlivých aktivit, které uživatel vykoná. Rozšířením logů přístupů, o data z mobilní aplikace, zaručuje přesnější vypovídající hodnotu výsledných analýz.

  Navrhuji zachování obdobných názvů jednotlivých aktivit. Pro snadnější orientaci v logu přístupů.

  Příklad: Student klikne na zobrazení rozvrhu.

  Ve webové aplikaci se do logu uloží URL:

  `/Pages/Student/Rozvrh.aspx`

  V aplikaci můžeme logovat jako:

  `Rozvrh`

  Dále můžeme přidat identifikátor zařízení. Mobilní aplikace, webová aplikace.

  Logování dat z mobilní aplikace nám napomůže k porozumění chování uživatelů v systému. Tímto nebudeme ochuzeni o důležitá data z mobilní aplikace.

- **Logování session**
  Po analýze a konzultaci s vedoucím diplomové práce, bylo navrženo logovaní session, které by přesněji vystihovalo chování uživatelů.

  Jelikož „session management“ není doposud v záznamech přístupů dostupný, nedokážeme tedy jednoznačně určit, kdy uživatel opustil webovou aplikaci a kdy se vrátil. Můžeme

pouze odhadovat a určovat začátek jednotlivých session, podle aktivity “výběr školy“, která se vykonává po přihlášení do systému.

Navrhl jsem tedy logování session, které by pomohlo u určování transakcí pro asociační pravidla (viz kapitola 3.3).

Výsledný log rozšířený o logování session a dat z mobilní aplikace by mohl vypadat takto:

”UserId”: 2
”RoleId”: 5
”SkolaInfoId”: 12
”KurzInfoId”: 80
”Cas”: ”2013-09-10T23:31:01.927”
”URL”: ”/Pages/Default.aspx”
”IP”: ”192.168.1.1”
”OS”: null
”SessionID”: ”P7omOxlpAh/CF6A2LGielX”
”Platform”: ”web application”

5 Analýza a návrh požadavků

Hlavním cílem aplikace je hlubší analýza chování uživatelů v systému. Zajímá nás, jak zlepšit uživatelskou přívětivost systému eLogika a jaké vztahy dokážeme v této analýze objevit.

Typické otázky můžou být: *„Dokáže chování studentů určit jejich výsledné hodnocení v kurzu?“*, *„Dokážeme podle chování studentů v minulých letech, předpovědět výsledky studentů v letech jiných?“*. Na tyto otázky se pokusíme odpovědět v následujících podkapitolách.

Tato kapitola je zaměřena na analýzu a návrh specifických požadavků, které jsou součástí zadaní diplomové práce.

5.1 Vizualizace chování uživatele v systému

Vizualizace chování uživatelů v eLogice, má sloužit jako nástroj pro zpětné dohledání historie návštěvnosti v systému. U každého uživatele eLogiky, bude aplikace umožňovat přehledné procházení jeho aktivit prováděných v jednotlivých dnech.

Tato funkce je primárně určena pro řešení krajních situací. Například student se dostane na stránky, ke kterým nemá přístup. Může se jednat o chybu systému, kterou je potřeba nutně odstranit. Tyto a další situace můžeme odhalit na základě prohlédnutí záznamů z námi vybraných dnů.

Požadavky:

- výběr uživatele podle příjmení nebo loginu v systému
- filtrovaní logů přístupů podle zadaného data (od – do)
- detailní zobrazení aktivit v jednotlivých dnech (čas, URL, IP adresa atd.)

K vizualizaci chovaní uživatele v systému nám postačí jeho logy přístupů. Záznamy bude zapotřebí rozdělit do kategorií podle jednotlivých dnů. V takto rozdělených kategoriích, bude možnost detailnějšího zobrazení aktivit, ke kterým uživatel přistupoval během dne.

Dále budeme využívat metodu *GetUserIdBySchoolsLogin* (viz kapitola 4.1), která nám podle zadaného loginu vrátí identifikátor uživatele, potřebný jako vstup pro metodu *getlogaccessbyiduzivatel* (viz kapitola 4.1), která vrací seznam všech logů uživatele.

5.2 Doporučování rozmístění jednotlivých prvků

Doporučování rozmístění jednotlivých prvků (webových stránek) v systému eLogika, hraje důležitou roli při zlepšování uživatelské přívětivosti systému.

Tato analýza by měla ukázat, na kterých stránkách uživatelé systému tráví nejdéle času. U těchto stránek můžeme případně doporučit posunutí na viditelnější místo v systému. Obecně se

může stát, že máme stránku, která je zanořená v systému pod několika odkazy, ale uživatelé ji hodně často navštěvují.

Cílem systému eLogika není, aby student zbytečně trávil čas hledáním důležité informace, ale aby nejdůležitější stránky byli co nejsnadněji dosažitelné.

Jedná se tedy o nástroj, který bude analyzovat konkrétní stránky v systému. Na základě analýzy odhalí „důležité“ stránky.

Požadavky:

- uživatel by měl mít možnost nastavit filtr minimálního procenta četnosti, pod kterou nelze považovat stránku za „důležitou“
- zobrazení grafu četností a průměrného stráveného času pro jednotlivé aktivity
- zobrazení tabulky četností a průměrného stráveného času pro jednotlivé aktivity

Jako hlavní zdroj informací budeme samozřejmě používat logy jednotlivých studentů. Důležitost daných stránek, budeme určovat podle průměrného času stráveného na aktivitě a četnosti zobrazení.

Tyto informace dohromady prezentujeme na grafu a seřadíme podle vypočtené důležitosti.

Administrátor systému si podle poskytnuté informace, může libovolně reorganizovat vnitřní strukturu samotného systému.

5.2.1 Určování důležitosti

Nyní si popíšeme, jakým způsobem budeme určovat důležitost stránky. Zavedeme si tedy dva pojmy, které budeme používat. Pojem „(ne)důležitá aktivita“ a „průchozí aktivita“.

1. **Průchozí aktivita**
   Průchozí aktivita je taková aktivita, která nastává poměrně často, ale uživatel se na ní dlouho nezdrží (tudíž stránku pouze prochází).

   Příklad: Máme-li průměrný čas strávený na všech aktivitách 15 sekund a průměrný počet zobrazení 1000x. Pak průchozí aktivitou je myšlena aktivita, která má například průměrný počet zobrazení 1500x a průměrný čas strávený na všech aktivitách 2 sekundy.

2. **Důležitá aktivita**
   Důležitá aktivita je taková aktivita, na které se uživatel delší dobu zdrží, ale uživatelé jí nenavštěvují často. Důležitá je tedy z pohledu analýzy. Tyto aktivity budou mít největší prioritu, protože se může jednat o stránky zanořené v systému, které uživatel nemůže najít.

Příklad: Aktivita, která má průměrný počet zobrazení 500x a průměrný čas strávený na všech aktivitách 20 sekund.

3. **Nedůležitá aktivita**
   Aktivita, na které se uživatel nezdrží dlouho nebo její průměrný počet zobrazení je zanedbatelný. Tyto aktivity je potřeba filtrovat a popřípadě odstranit ze systému, jedná-li se o nadbytečné stránky.

   Příklad: Aktivita, která má průměrný počet zobrazení 20x a průměrný čas strávený na všech aktivitách 3 sekundy.

Bude potřeba spočítat jednotlivý čas strávený na aktivitě. Dále spočítáme počet četností individuálních aktivit. Tyto informace dohromady prezentujeme na grafu a seřadíme podle vypočtené důležitosti.

Tento nástroj pouze ukazuje na stránky, které jsou hodně používané, a uživatel na nich tráví dostatek času. Tyto pak řadí podle důležitosti, čili poměr mezi četností a průměrnou dobou strávenou nad stránkou.

Administrátor systému si pak podle této informace, může libovolně reorganizovat vnitřní strukturu samotného systému.

5.3 Zjišťování chování uživatelů

Abychom mohli zjišťovat chování, měli bychom si nejprve vybrat, jakou skupinu uživatelů budeme analyzovat. Samozřejmě nemůžeme zvolit jen tak libovolné uživatele a prohlásit je za analyzovanou skupinu. Bude se tedy jednat o studenty ve stejném ročníku studia nebo v jednotlivých třídách. Pro tyto skupiny analyzujeme jejich chování. Podle zjištěného chování je dále dělíme do uměle vytvořených podskupin neboli shluků. Jednotlivé shluky jsou charakteristické tím, že obsahují studenty s podobným chováním v systému.

Zjišťování chování uživatelů v systému budeme analyzovat za pomoci shlukovacích metod (viz kapitola 3.2). Rozdělením uživatelů podle chování a následovného ohodnocení vytvořených skupin, budeme zkoumat existenci vztahu, mezi určitým typem chování a výsledným hodnocením kurzu. Pokud potvrdíme existenci tohoto vztahu, můžeme analyzovat chování uživatelů a předpovědět, jestli mají tendenci spadat spíše do kategorie prošel nebo neprošel.

Požadavky:

- určování chování uživatelů v jednotlivých třídách
- určování chování uživatelů ve školním roce (jednoho předmětu)
- rozdělení uživatelů do skupin podle podobnosti chování

- zjištění existence vztahu mezi určitým chováním a výsledným hodnocením

Zvolíme si tedy skupinu uživatelů, kterou chceme analyzovat. Pod skupinou máme na mysli všechny studenty v jednom školním roce, nebo v jedné třídě.

Provedeme shlukovou analýzu nad jejich chováním a získáme skupiny uživatelů, kteří jsou svým chováním podobní. Na tyto uměle vytvořené skupiny, aplikujeme výpočet statistických hodnot (jako je například průměrné hodnocení v kurzu). Tímto zjistíme, zda chování uživatelů ovlivňuje výsledné hodnocení studentů.

5.4 Ostatní požadavky

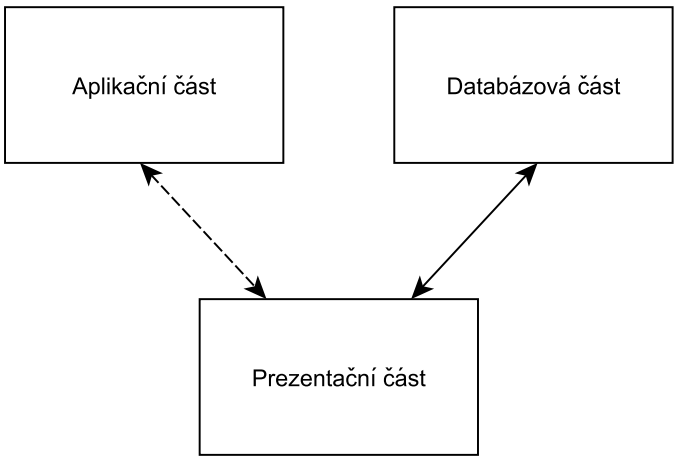
Zde si uvedeme seznam specifických požadavků, které nejsou definovány v zadání, ale byli navrhnuty vedoucím práce.

- **Univerzální platforma** – jedním ze specifických požadavků je aplikaci vyvinout v UWP platformě. Tato platforma vznikla s nástupem Windows 10. Pomocí UWP dokážeme vytvořit jednu aplikaci pro vícero zařízení (mobil, tablet, pc atd.) (více v kapitole 6.5).
- **Aplikace na Windows Store** – tento požadavek je úzce spjatý s vývojem na univerzální platformě. Tyto aplikace se distribuují na Windows Store [20]. Aplikace tedy bude nahrána na Windows Store vedoucího diplomové práce.
- **Znovu použitelnost** – tento požadavek klade důraz na využitelnost již naprogramovaných metod a funkcí pro další vývojáře pracující na systému eLogika. Proto byla logika aplikace přesunuta do separátní PCL knihovny (více v kapitole 6.5), díky ní můžeme metody používat i na desktopech/konzolových aplikacích, mobilních atd.

Jedná se spíše o funkční požadavky, protože aplikace je určená primárně pro vedoucího této práce, bylo je nutno zahrnout do samotného vývoje aplikace.

5.5 Struktura aplikace

Základním konceptem aplikace bude snaha oddělit jednotlivé části systému. Aplikace byla rozdělena do tří částí. Část aplikační (výpočty, algoritmy, komunikace ze serverem), databázová (uložiště) a prezentační (GUI). Rozdělení aplikace usnadňuje znovu použití jednotlivých prvků.



Obrázek 7: Rozdělení částí aplikace

V této kapitole si tedy popíšeme jednotlivé části, a jak společně budou tvořit celistvou aplikaci.

5.5.1 Aplikační část

Jedním z požadavků, které byly kladeny na výslednou aplikaci, je možnost jejího rozšíření nebo znovu použitelnosti v budoucnu (viz 5.4). Většina funkcí aplikace byla navržena právě tak, aby splňovala tento požadavek.

Aplikační část neboli framework je základním funkčním prvkem celé aplikace. Oddělením logické části do separátní DLL knihovny, která bude obsahovat většinu funkční části aplikace (výpočty, přepočty, algoritmy, práci a komunikaci se serverem systému eLogika), docílíme požadovaného výsledku.

Přerušovaná čára v obrázku 7 naznačuje její oddělení od aplikační části. Ta je zcela samostatná a není závislá na databázové ani prezentační části. Její nezávislost nám umožnuje její znovu použitelnost pro jiné aplikace.

Aplikační část programu obsahuje tyto komponenty:

- komunikace ze serverem eLogika
- vybrané data miningové metody

- složené metody se skládají z více dotazů na server eLogiky

Implementace aplikační části je detailněji popsaná v kapitole (viz 6.1).

5.5.2 Databázová část

Velká část analýz bude prováděna nad daty z předešlých ročníků. Tyto data jsou v systému neměnná.

Využití má při ukládání dat o jednotlivých kurzech. Typicky data o uživatelích a jejich aktivitách. Tyto informace jsou serializovány do jednoho XML souboru a dále používány v aplikaci, pro zrychlení celkového načítání. Pracujeme s předpokladem, že u již proběhnutých kurzů(tříd), se data v průběhu dalších let nemění. Počet studentů v rámci kurzu zůstává stejný, jejich aktivity za dané období také atd.

Další využití najdeme při ukládání popisu jednotlivých aktivit. Systém eLogika se stále rozšiřuje a přidávají se do něj nové funkce. Jednotlivé aktivitě (stránce) v systému, můžeme dát určitý název, abychom ve výsledných analýzách měli větší přehled, o jakou analyzovanou aktivitu se jedná. Typicky například `/Pages/Default.aspx` si můžeme popsat jako *hlavnistranka*.

Databázovou částí je tedy myšleno interní úložiště pro uchovávání výsledku analýz a pojmenovávání jednotlivých aktivit.

Návrh tabulek pro uchovávání dat:

Obrázek 8: Databázová část - návrh tabulek pro uchovávání dat

- **GlobalStudentUrlTable** – struktura tabulky pro pojmenování jednotlivých aktivit. Obsahuje *Url* aktivity a tu přejmenuje na *Title*. Dále můžeme přidat i popis dané aktivity.
- **SxAMatrixDownloadedTable** – návrh tabulky, která uchovává XML data o již stažených datech. Obsahuje datum stažení dat, parametry pro specifikaci skupiny (škola, kurz, rok atd.) a jméno souboru, do kterého se data ukládala.

Implementace databázové části je detailněji popsaná v kapitole (viz 6.2).

5.5.3 Prezentační část

Důležitou součástí aplikace je samotná její prezentace. V prezentační části se budeme snažit, aby vzhled byl příjemný a ovládání uživatelsky přívětivé. Aplikace bude vyvíjena na platformě UWP(viz kapitola 6.5), takže bude potřeba její funkčnost otestovat i na ostatních podporovaných zařízeních. V prezentační části budeme provádět také zobrazování grafů a tabulek.

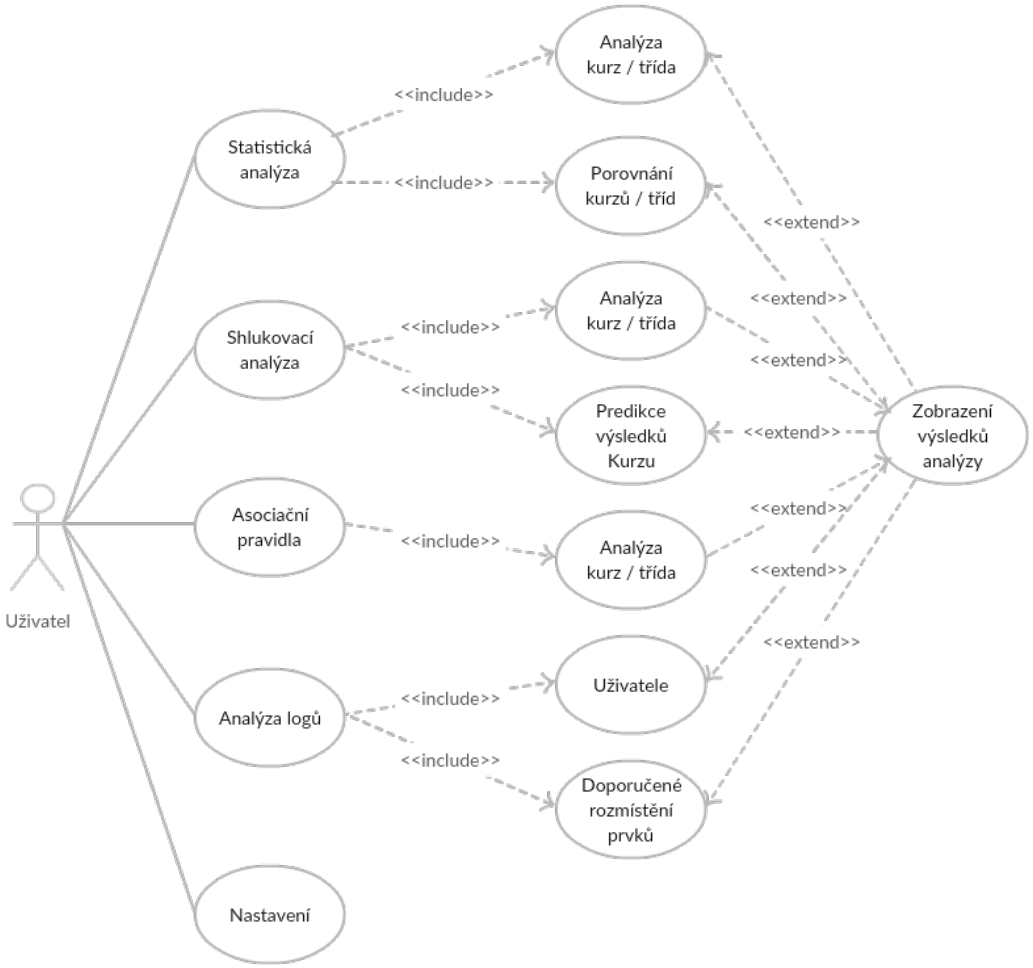
Aplikace bude rozdělena do jednotlivých sekcí:

- Statistická analýza
- Shlukovací analýza
- Asociační pravidla
- Analýza logů
- Nastavení

Pro jednotlivé sekce bude vždy zobrazen výčet analýz, které může uživatel spustit. Například u statistické analýzy, můžeme analyzovat jednotlivé kurzy. Pro shlukovací analýzu máme například predikci výsledků (viz obr. 9).

Pro každou takto vybranou analýzu se nám zobrazí nové okno s nastavením příslušných parametrů (analyzovaný ročník, počet shluků atd.).

Případy užití aplikace:



Obrázek 9: Diagram případů užití aplikace

Implementace prezentační části je detailněji popsaná v kapitole (viz 6.3).

6 Implementace

V této kapitole si detailněji popíšeme implementaci analyzovaných a navrhnutých postupů. Ukážeme části zdrojových kódů některých řešení a algoritmů.

6.1 Aplikační část

Tato část byla oddělena do samostatné knihovny PCL (6.5.2) s názvem *elcorelib.dll*. Díky přesunutí všech metod do separátní přenosné knihovny, můžeme jednotlivé funkce spouštět například v konzolové aplikaci, UWP ,WPF a ostatních platformách.

Elcorelib knihovna obsahuje tyto komponenty (namespaces):

- **API** – obsahuje přímou komunikaci ze serverem eLogika
- **DataMatrix** – obsahuje datovou matici, ve které se vypočítávají data pro určitou skupinu uživatelů
- **DataMining** – obsahuje vybrané data miningové metody a algoritmy (asociační pravidla, shlukovaní atd.)
- **Methods** – obsahuje wrapper nad komunikací ze serverem, tzn. zpracovává data ze serveru eLogika, do určité podoby
- **Utilities** – obsahuje užitečné nástroje (měření vzdáleností, serializace atd.)

6.1.1 DataMatrix namespace

Tento namespace slouží k práci s vícerozměrnými maticemi, které uchovávají informace o jednotlivých skupinách uživatelů v systému.

Hlavní třídou je zde *SxAMatrix*, kterou si popíšeme podrobněji.

Třída SxAMatrix
Jedná se o matici [student x aktivita] viz obr.10, využívanou pro uchovávání a výpočet dat pro vybranou skupinu uživatelů.

Metoda *DownloadData* viz 1, stáhne ze serveru eLogika jednotlivé studenty v zadaném kurzu/třídě a jejich logy přístupů v období studovaného semestru. Tyto data pak přepočítá do matice ve formátu, kde u každého studenta jsou četnosti jednotlivých aktivit.

```
private void DownloadData(IProgress<int> prog = null)
{
   ClearInitData();
   ...
   var progress = 0;
   foreach (var student in studentsInCourse)
   {
      var st = new Student { Id = student.UserId, FirstName =
          student.FirstName, LastName = student.LastName};
      SetStudentScore(st);
      SetStudentActivities(st, SchoolClassInfoModel.RoleId);
      ...
      var calc = (double) ++progress / studentsInCourse.Count * 100;
      prog?.Report((int)calc);
   }
   ...
}
```

Výpis 1: SxAMatrix - metoda DownloadData

Metoda *ToDoubleArray*, vrací matici ve tvaru [student x aktivita] viz obr.10, která se předává ostatním data miningovým algoritmům ke zpracování.

$$\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ \ldots \\ S_m \end{array} \begin{array}{c} \begin{array}{cccccc} A_1 & A_2 & A_3 & A_4 & \ldots & A_n \end{array} \\ \begin{pmatrix} 12 & 29 & 3 & 1 & \ldots & 0 \\ 42 & 36 & 8 & 3 & \ldots & 0 \\ 6 & 3 & 9 & 2 & \ldots & 2 \\ \ldots & \ldots & \ldots & \ldots & .. & .... \\ 38 & 20 & 36 & 5 & \ldots & 0 \end{pmatrix} \end{array}$$

Obrázek 10: SxAMatrix - student x aktivita

$S_1 \ldots S_m$ – řádky matice určují jednotlivé studenty
$A_1 \ldots A_n$ – sloupce určují četnosti jednotlivých aktivit

6.1.2 DataMining namespace

Zde jsou obsaženy všechny data miningové a statistické metody, které využíváme pro analýzu.

- Statistické metody (viz 3.1)
- Shlukovací metody (viz 3.2)

- Asociační pravidla (viz 3.3)

Většinu popsaných metod již obsahuje analytická knihovna ALGLIB (viz 6.4.4). Popíšeme si tedy některé metody a algoritmy.

- **Statistické metody**
  Veškeré statistické metody používané k naší analýze (viz 3.1) jsou dostupné v knihovně ALGLIB (viz 6.4.4). Uvedeme si pro ilustraci volaní metod pro aritmetický průměr a rozptyl.

  Aritmetický průměr:

```
public static double Mean(double[] x)
{
   return alglib.samplemean(x);
}
```

Výpis 2: ALGLIB - aritmetický průměr

  Rozptyl:

```
public static double Variance(double[] x)
{
  return alglib.samplevariance(x);
}
```

Výpis 3: ALGLIB - rozptyl

  Na vstupu metody očekávají soubor zkoumaných hodnot, výstupem je požadovaný průměr, rozptyl, medián atd.

- **Shlukovací metody**
  Knihovna ALGLIB (viz 6.4.4) obsahuje i některé metody pro shlukování, například hierarchické shlukování (AHC) a metodu K-means.

  Metoda K-means:
  Metoda K-means přijímá parametr K, který určuje výsledný počet shluků. Následně pak matici jednotlivých vektorů, které chceme takto rozdělit. Výstupem je pole hodnot v rozsahu 0 – (K-1), které určuje jaký vektor má být zařazen do kterého shluku.

```
public static int[] KMeans(int k, double[,] matrix)
{
   alglib.clusterizerstate s;
   alglib.kmeansreport rep;
   alglib.clusterizercreate(out s);
   alglib.clusterizersetpoints(s, matrix, 2);
   alglib.clusterizersetkmeanslimits(s, 20, 10);
   alglib.clusterizerrunkmeans(s, k, out rep);
   return rep.cidx;
}
```

Výpis 4: ALGLIB - metoda K-means

- **Asociační pravidla**
  Knihovna ALGLIB neobsahuje asociační pravidla, proto je musíme implementovat. Více o asociačních pravidlech v kapitole 3.3.

  Algoritmus Apriori:
  Apriori algoritmus oproti triviálnímu vyhledává v datech frekventované množiny položek (ty, které mají alespoň minimální stanovenou podporu). Z této množiny pak generuje silné asociace (takové, které ještě navíc dosahují i minimální spolehlivosti).

  Vstupem algoritmu jsou jednotlivé transakce, zadaná minimální spolehlivost a podpora. Prvním krokem je vyhledání všech frekventovaných kombinací atributů, které splňují zadanou minimální podporu.

  V dalším kroku se vyberou pouze frekventovaná pravidla, která splňují i minimální spolehlivost. Tímto krokem dostaneme takzvaná silná pravidla. (více o algoritmu apriori v literatuře [21] a [22]).

6.1.3 API namespace

Komunikace ze serverem systému eLogika. V návrhu aplikace (viz 5.5) jsme rozhodli, že samotná komunikace se serverem bude součástí hlavní funkční knihovny.

Je tedy obsažena v knihovně *elcorelib.dll*, ve jmenném prostoru *elcorelib.API* ve statické třídě *ApiRequest*.

Tato třída má za úkol, provádět veškerou komunikaci se serverem eLogiky. Stahovat data z jednotlivých služeb a převádět je do připravených modelů. Třída obsahuje metody pro získání (GET) a odeslání (POST) dat na server.

- **Metoda GET**
  Metoda *GetData* vytvoří dotaz pomocí *HTTPWebRequest* třídy na URL a vrací response typu *HTTPWebResponse*, u které můžeme přečíst celou odpověď.

```
private static string _GetData(String urlrequest)
{
   int request = (HttpWebRequest) WebRequest.Create(urlrequest);
   var response = request.BeginGetResponse(null, request);

   HttpWebRequest myRequest = (HttpWebRequest)response.AsyncState;
   using (HttpWebResponse myResponse =
       (HttpWebResponse)myRequest.EndGetResponse(response))
   {
      using (StreamReader httpwebStreamReader = new
          StreamReader(myResponse.GetResponseStream()))
   {
   return httpwebStreamReader.ReadToEnd();
}
```

Výpis 5: ApiRequest - metoda GetData

- **Metoda POST**
  Metoda post, odesílá data pomocí asynchronní metody *PostAsync* na server ve formátu JSON.

```
public static async Task<string> _PostData(string urlrequest, string
    jsonstring)
{
   HttpContent contentPost = new StringContent(jsonstring, Encoding.UTF8,
       "application/json");
   HttpClient client = new HttpClient();
   HttpResponseMessage re = await client.PostAsync(urlrequest,
       contentPost);

   return await re.Content.ReadAsStringAsync();
}
```

Výpis 6: ApiRequest - metoda PostData

- **Metody**
  Objasníme si implementaci potřebných metod k analýze, které jsme si popsali v kapitole (viz 4.1.2)

  Kód každé metody se skládá ze dvou částí:

  - **Soukromá (private) metoda** – přidá k URL hodnoty parametrů a vrací odpověď serveru v textové formě. (viz kod 7)
  - **Veřejná (public) metoda** – deserializuje příslušnou textovou odpověď na objekt. (viz kod 8)

```
private static string _GetLogAccess(int userId, string from, string to)
{
    return _GetData(string.Format(UrlApiGetlogaccessbyiduzivatel, userId,
        from, to));
}
```

Výpis 7: Private metoda pro GetLogAccess

Ve veřejné metodě už probíhá samotné parsování JSON řetězce na daný objekt. JsonConvert je metoda z knihovny JSON.NET popsaná v kapitole 6.4.5.

```
public static List<LogAccessModel> GetLogAccess(int userId, string from,
    string to)
{
   return JsonConvert.DeserializeObject<List<LogAccessModel>>
   (
      _GetLogAccess(userId,from,to)
   );
}
```

Výpis 8: Public metoda pro GetLogAccess

Jak můžeme vidět, tahle metoda vrací List<*LogAccessModel*>, kde třída *LogAccessModel* je příslušný model (viz obr. 9) v JSON odpovědi, kterou vrací server na daný dotaz. V tomto případě model logu přístupů, který obsahuje *UserId*, *RoleId*, *CourseInfoId* atd. Záznamy přístupu byly popsány v kapitole (viz 4.2).

- **Model**

```
[JsonObject]
public class LogAccessModel
{
    [JsonProperty(PropertyName = "UserId")]
    public int UserId { get; set; }

    [JsonProperty(PropertyName = "RoleId")]
    public int RoleId { get; set; }

    [JsonProperty(PropertyName = "KurzInfoId")]
    public int CourseInfoId { get; set; }

    [JsonProperty(PropertyName = "Cas")]
    public DateTime DateTime { get; set; }
    [JsonProperty(PropertyName = "Url")]
    public string Url { get; set; }

    [JsonProperty(PropertyName = "Ip")]
    public string Ip { get; set; }
}
```

Výpis 9: Model - LogAccessModel

Třída *ApiRequest* tedy obsahuje všechny poskytnutá data vypsána v kapitole (viz 4.1), pro každou službu bylo potřeba vytvořit její objektový model a metodu, která jí zpracovává.

6.2 Databázová část

Implementace databázové části, stejně tak aplikační část je vytvořena jako samostatná PCL knihovna. Tímto jsme zachovali možnou rozšiřitelnost a oddělitelnost jednotlivých podsystémů. Jedná se tedy o jednoduchou knihovnu se schopností vytvářet, zapisovat a mazat jednotlivá data, která dle návrhu aplikace (viz 5.5) budeme potřebovat. Knihovna používá pro práci s lokální databází rozšíření SQLite.NET (viz 6.4.2).

Knihovna nazvaná *datastoragelib.dll*, obsahuje:

- třídu *Database* pro operace s lokální databází (připojení, vytváření, editace, mazání dat).
- statickou třídu *LocalStorage*, za pomocí které můžeme kdekoliv v aplikaci získat potřebná uložená data.

6.2.1 Třída Database

Příklad vytvoření databázového souboru:

```
public Database(ISQLitePlatform platform, string folderPath, string fileName =
    "db.sqlite")
{
   _platform = platform;
   _dbFilePath = System.IO.Path.Combine(folderPath, fileName);
   _connection = new SQLiteConnection(_platform, _dbFilePath);
}
```

Výpis 10: Třída Database - konstruktor

- *ISQLitePlatform* – definuje platformu, na které databázi vytváříme (Win32, WinRT)
- *folderPath* – cesta pro uložení databázového souboru
- *fileName* – název databázového souboru, výchozí název *db.sqlite*

Příklad vytvoření tabulky:

```
public int CreateTable(Type type)
{
   using (_connection = new SQLiteConnection(_platform, _dbFilePath))
   {
      return _connection.CreateTable(type);
   }
}
```

Výpis 11: Třída Database - metoda CreateTable

Příklad vložení záznamu do tabulky:

```
public int Insert(object obj)
{
   using (_connection = new SQLiteConnection(_platform, _dbFilePath))
   {
      return _connection.Insert(obj);
   }
}
```

Výpis 12: Třída Database - metoda Insert

6.2.2 LocalStorage.cs

V této třídě najdeme metody, které ukládají jednotlivé analýzy, mažou záznamy a editují. Pro příklad si uvedeme metodu, která kontroluje, zda analyzovaná skupina již existuje.

```
public static bool SxAIsExist(string calledParameter)
{
   return
   _db.GetAll<SxAMatrixDownloadedTable>().Exists
   (
      x=>x.CalledParameters.Equals(calledParameter)
   );
}
```

Výpis 13: LocalStorage - metoda SxAIsExist

calledParameter – parametry analýzy (schoolId, courseInfoId atd.)

6.3 Prezentační část

V prezentační části jsem se snažil vytvořit vzhled aplikace tak, aby byl jednoduchý, uživatelsky přívětivý a působil konzistentně. Proto jsou jednotlivé analýzy v podobném rozložení, a uživatel si tak jednoduše zvykne na jejich styl. Prezentační část je napsána v jazyku XAML (viz 6.5.3) a při její implementaci jsem se řídil vzorem MVVM (viz 6.5.4).

6.3.1 Výběr analýzy

Hlavní menu je umístěno v horní části aplikace, kde si zvolíme požadovaný typ analýzy (statistickou, shlukovací, asociační pravidla atd.).

Jak jsme mohli vidět na případech užití (viz obr. 9), každý typ analýzy má svou určitou skupinu, na níž můžeme analýzu provést (kurz, třida atd.). Jednotlivé skupiny můžeme vybírat pomocí „tlačítka“ SfHubTile , které nás přesměruje na zvolenou analýzu.

```
<PivotItem>
   <controls:WrapPanel Style="{StaticResource PivotItemPanels}">
      <notification:SfHubTile Title="Shluky" Header="Kurzy"
         Style="{StaticResource HubStyle}" Command="{Binding
         NavigateToClCourseCommand}" />
      ...
```

Výpis 14: Prezentační část XAML - SfHubTile

6.3.2 Analýza

Stránka s analýzou většinou obsahuje menu výběru. Jednotlivé analýzy jsou samostatná view ve vzoru MVVM (viz kapitola 6.5.4). Ukážeme si zde pouze kód propojení view a view modelu.

```
<paging:MtPage.Resources>
   <viewModel:StClassViewModel x:Key="ViewModel" />
</paging:MtPage.Resources>

<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"
    DataContext="{StaticResource ViewModel}">
   <Grid.RowDefinitions>
      ...
```

Výpis 15: Prezentační část XAML - propojení view a view model

6.3.3 Menu výběru

V menu si vybíráme kurz nebo třídu, kterou chceme analyzovat.

```
   <navigation:SfTreeNavigator ItemsSource="{Binding SelectorList}"
                     SelectedItem="{Binding SelectedSubject,
                     UpdateSourceTrigger=PropertyChanged, Mode=TwoWay}"
                     Style="{StaticResource SelectMenuStyle}">
      ...
   </navigation:SfTreeNavigator>
</Grid>
```

Výpis 16: Prezentační část XAML - menu výběru SfTreeNavigator

6.3.4 Indikátory

Indikátory poskytují zpětnou vazbu uživateli o průběhu zpracování časově náročných operací.

V aplikaci se nachází dva typy indikátorů průběhů:

- *ProgressBar* – Zobrazujeme v situacích, kdy předem známe délku zpracování požadavku. Konkrétně se může jednat o zpracovávání všech studentů v kurzu, u kterého známe jejich počet. Snadno pak dokážeme odhadnout procento zpracovávaného požadavku.
- *Busy indicator* – Indikátor zaneprázdněnosti používáme v situacích, u kterých nedokážeme odhadnou jak dlouho bude zpracovávaný proces trvat.

6.3.5 Ikony

Pro ikony do aplikace jsem použil Metro studio[1], které je dostupné i jako rozšíření do Visual Studia. Metro studio je software zdarma od společnosti Syncfusion (viz 6.4.6), který obsahuje více než 6 tisíc volně dostupných ikon. Tyto ikony lze libovolně modifikovat, zvětšovat, zmenšovat měnit barvu, pozadí.

6.4 Použité knihovny

Seznam všech použitých knihoven:

- SQLite.NET
- MyToolkit
- Alglib2
- Json.NET
- Syncfusion

Všechny knihovny, které byly použity pro aplikaci jsou v NuGet repozitáři. V krátkosti si tedy přiblížíme NuGet. Následně přistoupíme k samostatnému popisu jednotlivých knihoven a k jakému účelu nám poslouží.

6.4.1 NuGet

Rozšíření NuGet je open-source správce jednotlivých knihoven pro vývoj aplikací pod Microsoftem. V dnešní době často používán .NET vývojáři a to především díky integraci do Visual Studia. Tento chytrý správce knihoven dokáže jednoduše přidávat knihovny do projektů.

Jednotlivé knihovny můžeme přidávat do projektu ve Visual Studiu přes *(Tools – NuGet Package Manager – Manage NuGet Packages for Solution)*. Alternativou přidávání knihoven do projektu je použití Package Manager Console,*(Tools – NuGet Package Manager - Package Manager Console)*, ve které můžeme přidávat knihovny pomocí *install-package* příkazů. Veškeré dostupné knihovny můžeme nalézt na oficiálních stránkách[2].

6.4.2 SQLite.NET

Knihovna pro práci s lokálními databázemi. Díky této knihovně dokážeme snadno z modelů vytvořit a naplnit databázi. Jedná se tedy o objektové relační mapování (ORM). Nemusíme psát jednotlivé SQL dotazy, ale postačí nám jednoduché metody typu *CreateTable* a předáme jí model, který definuje strukturu tabulky. Stejně to funguje s příkazy insert, delete, update.

[1] `https://www.syncfusion.com/downloads/metrostudio/`
[2] `https://www.nuget.org/`

Knihovna SQLite.NET[3] je použitá v databázové části (kapitola 5.5.2), která slouží pro uchovávání interních informací aplikace.
install-package příkaz: *Install-Package SQLite.Net-PCL*

6.4.3 MyToolkit

Rozšiřující knihovna[4] obsahující soubor .NET doplňků, pro platformu UWP a WinRT. Obsahuje užitečné MVVM třídy, které budeme používat v této aplikaci například *RelayCommand*, *ViewModelBase* a další.

install-package příkaz: *Install-Package MyToolkit*

6.4.4 ALGLIB

Analytická knihovna ALGLIB[5], která obsahuje velké množství užitečných metod.
Obsahuje například funkce pro:

- Analýzu dat
- Lineární algebru

Alglib je dostupný v edici zdarma pod licencí GPL.

install-package příkaz: *Install-Package alglibnet2*

6.4.5 Json.NET

Je open-source knihovna pro práci s JSON. Tuto knihovnu používám, protože komunikace se serverem eLogika probíhá ve výměně JSON řetězců. Využití najdeme například v serializaci a deserializaci JSON řetězců.

install-package příkaz: *Install-Package Newtonsoft.Json*

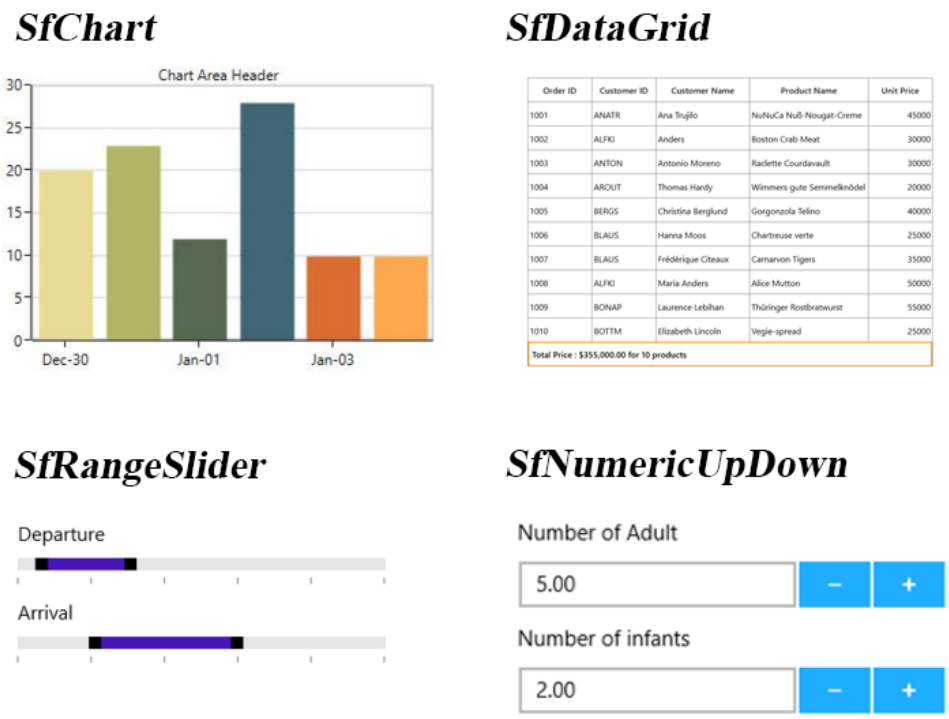
6.4.6 Syncfusion

Pro grafy, tabulky a některé další UWP komponenty, jsem použil komunitní verzi nástrojů od firmy Syncfusion. Jedná se o nástroj více než 35 doplňujících komponent, které nejsou obsaženy jako základní komponenty ve Visual Studiu. Dokumentace k jednotlivým komponentám UWP je dostupná na oficiálních stránkách[6].

[3] `http://www.sqlite.org/`
[4] `https://github.com/MyToolkit/MyToolkit`
[5] `http://www.alglib.net/`
[6] `http://help.syncfusion.com/uwp/overview`

Použité komponenty Syncfusion:

- *SfChart* – komponenta pro vykreslování různých druhů grafů
- *SfDataGrid* – komponenta pro zobrazování složitějších tabulek. Obsahuje automatické třídění atd.
- *SfNumericUpDown* – komponenta, která umožňuje inkrementaci čísel pomocí + a − tlačítek. Výborné pro aplikace podporující dotykové gesta
- *SfBusyIndicator* – tato komponenta zobrazuje indikátor, pokud je aplikace vytížená a pracuje se složitějším výpočtem na pozadí
- *SfTreeNavigator* – menu ve stromové struktuře, umožňuje přehlednou navigaci a výběr podsekcí
- *SfTextBoxExt* – komponenta TextBox rozšířená o zajímavé funkce (vodoznak místo textu)
- *SfRangeSlider* – komponenta pro určení intervalu hodnot.



Obrázek 11: Ukázka Syncfusion komponent

Jejich licence je zdarma pro nekomerční účely a do hrubého ročního obratu nižšího než jeden milion Amerických dolarů. Což je v našem případě v pořádku a můžeme tyto komponenty používat.

6.5 Použité technologie

V této podkapitole si popíšeme jednotlivé technologie a vzory, které jsem použil při implementaci jednotlivých částí aplikace. Popíšeme si detailněji platformu, na které výsledná aplikace funguje.

6.5.1 Universal Windows Platform

S nástupem nového operačního systému Windows 10 byla vydána nová platforma Universal Windows Platform (UWP). Tato platforma je zajímavá, protože přináší nové unifikované jádro, díky kterému máme možnost spouštět jednu aplikaci na různých Windows 10 zařízeních (mobil, tablet, pc, xbox atd.) [23].

Obrázek 12: Universal Windows Platform[7]

Jedná se o nástupce Windows Runtime (WinRT) platformy, která byla představena ve Windows 8. Aplikace pro tuto platformu jsou primárně určeny pro Windows Store, který je nabízí zařízením na kterých je aplikace schopná provozu.

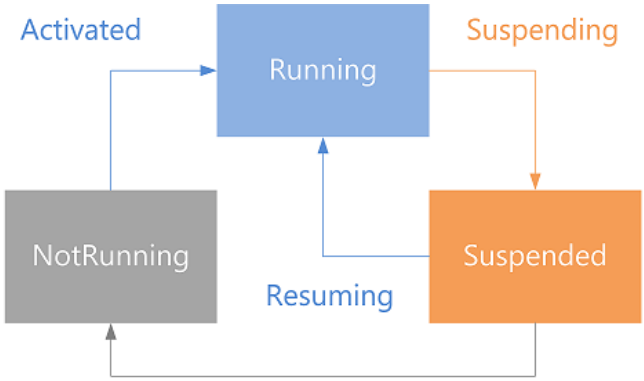
Jelikož je UWP aplikace schopná fungovat na různých zařízeních, je zapotřebí optimalizovat výslednou aplikaci pro jednotlivá zařízení v důsledku využití jejich potenciálu. Můžeme tedy napsat specifický kód pro odlišná zařízení, a tím například na mobilní platformě využít dotyková gesta, gyroskopický senzor a další funkce.

Životní cyklus UWP aplikace

Tato podkapitola popisuje životní cyklus aplikace univerzální platformy od spuštění aplikace po

[7] zdroj: `https://msdn.microsoft.com/en-us/windows/uwp/get-started/images/universalapps-overview.png`

její zavření. Protože uživatelé upřednostňují práci s několika aplikacemi najednou, očekávají, že aplikace si budou schopny pamatovat stav, ve kterém byly přerušeny (uživatel přepnul na jinou aplikaci). Měli bychom tedy porozumět jednotlivým stavům aplikace v životním cyklu.



Obrázek 13: Životní cyklus UWP aplikace[8]

V UWP aplikaci tedy z můžou nastat tyto stavy:

- *Running* – Aplikace, kterou uživatel spustí, se dostává ze stavu notrunning do stavu running po zobrazení úvodní obrazovky (splash screen).
- *Suspended* – Stav suspended nastane, kdykoliv uživatel přepne na jinou aplikaci. Dále se můžeme dostat do stavu suspended, pokud se zařízení dostane do režimu úsporu baterie. Jakmile se uživatel vrátí zpět, aplikace se načte do původního stavu.
- *NotRunning* – Poté co uživatel zavře aplikaci, nejprve se dostane do stavu suspended a poté se ukončí. V této chvíli se nachází ve stavu not running.

Více o jednotlivých stavech a přechodech mezi nimi se můžeme dozvědět na oficiálním webu Microsoftu [24].

6.5.2 Portable Class Library

Knihovny PCL nám umožňují vytvářet knihovny, které pracují na více než jedné .NET platformě. Můžeme vytvářet třídy obsahující kód, který chceme sdílet mezi několika různými projekty. Použitím PCL dokážeme vytvořit přenosnou knihovnu, která bude fungovat bez jakékoliv modifikace na .NET Framework, Silverlight, Windows Phone 7 nebo Xbox 360 platformě [25]. U PCL knihovny si můžeme vždy změnit nastavení, jaké platformy bude knihovna podporovat.

Pomocí PCL knihovny můžeme například testovat jednotlivé funkční části aplikace v konzoli, což nám ušetří čas.

[8] zdroj: `https://i-msdn.sec.s-msft.com/en-us/windows/uwp/launch-resume/images/state-diagram.png`

6.5.3 XAML

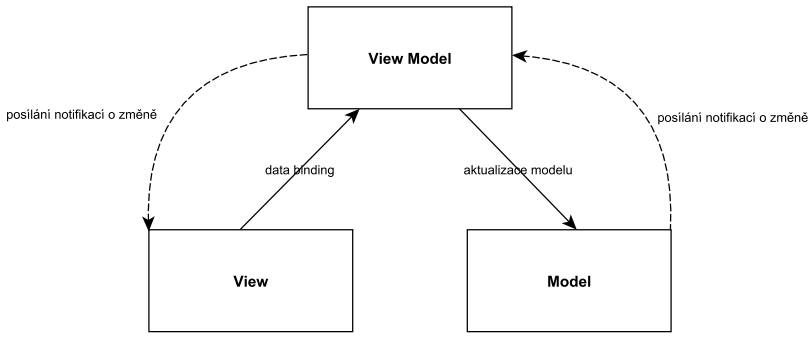
XAML je deklarativní jazyk na principu XML. V aplikacích UWP slouží k vytváření objektů a definování vzhledu aplikace. Tento jazyk byl zpopularizován díky WPF a Silverlight.

Jazyk XAML nám usnadňuje zápis kódu. Cokoliv napíšeme v XAML dokážeme napsat i pomocí kódu přímo do programu. Základem je oddělení prezentační části aplikace od funkční části. Popisujeme v něm objekty, které se mají vytvořit, jaké parametry se mají nastavit. Více v literatuře [20] a [26].

6.5.4 MVVM

Model-View-ViewModel vzor, který můžeme používat na všech platformách používajících XAML. Snahou tohoto vzoru je oddělit uživatelské rozhraní od vlastní logiky. Jak správně používat MVVM se můžeme dočíst na oficiálním webu MSDN Microsoft [27].

MVVM obsahuje tři základní komponenty. Model, view a view model. Každý z těchto komponent plní odlišnou roli. Vztah mezi jednotlivými komponenty si ukážeme na následujícím obrázku (obr. 14).



Obrázek 14: Komponenty MVVM

Abychom pochopili jak tyto komponenty fungují podrobněji si je popíšeme.

- **Model**
  Jedná se o samotná data s nimiž aplikace pracuje. Model ve vzoru MVVM nesmí obsahovat žádnou metodu, nebo funkční část. Slouží tedy pouze k uchovávání dat a zpětné notifikaci o jeho změně.

- **View**
  Obsahuje samotnou implementaci vzhledu uživatelského rozhraní. View je psáno jazykem XAML (viz 6.5.3) a obsahuje pouze definice a vlastnosti designu. Logickou část view tvoří samotný view model, který se spojuje přes DataContext. Jednotlivé ovládací prvky jsou napojeny pomocí bindingu s view modelem.

- **View Model**

  Ve view modelu je obsazena veškerá logika napojeného view. Udržuje se zde také stav aktuálního view. Jedná se tedy o prostředníka mezi modelem a view. View model plní důležitou roli, aktualizuje totiž samotné view. Například, uživatel stiskne tlačítko, tím vyvolá určitou metodu ve view modelu a ta adekvátně upraví view.

Výhody MVVM

Celá tato konstrukce nám umožňuje vytvářet aplikace, které jsou jednodušeji modifikovatelné. Například, máme finální verzi aplikace a chceme modifikovat vzhled. Bez jediného zásahu do vnitřního kódu nebo logiky nám stačí upravit pouze příslušné XAML view a máme hotovo.

Můžeme testovat view model a model bez závislosti na view. Detailněji popsaný vzor MVVM i s příklady použití najdete ve webovém tutoriálu na MSDN [28].

7 Data mining a výsledky analýzy

Po analýze dat, návrhu a implementaci všech částí programu se dostáváme ke konečnému kroku, a tím jsou výsledky jednotlivých analýz. Výsledky jsou rozdělené do podkapitol na výsledky statistické analýzy, shlukové analýzy a asociačních pravidel.

Jelikož si díky nastavování parametrů analýzy dokážeme generovat poměrně široké spektrum výsledků, uvedeme si pouze vybrané příklady, u kterých je zhodnotíme.

7.1 Výsledky statistické analýzy

Statistická analýza je rozdělena na analýzu kurzů, tříd a jejich porovnávání. Do této analýzy dále spadá doporučené rozmístění prvků, jelikož využívá statistických údajů používání stránek, pro jejich doporučení.

Uvedeme si tedy jeden případ analýzy porovnání kurzů a doporučeného rozmístění prvků a zhodnotíme si výsledky.

Příklad 1: Porovnání kurzů
Pro příklad uvedu analýzu předmětu Matematická logika za rok 2013/2014 prezenční studium a porovnáme s rokem 2014/2015 také prezenční studium.

	ML 2013/2014	**ML 2014/2015**
Počet studentů:	**143**	141
Průměrné body za zápočet:	17.09b	**20.37b**
Průměrné body za zkoušku:	**28.75b**	26.57b
Průměrné body celkem:	45.84b	**46.93b**
Předmět úspěšně dokončilo:	**82 studentů**	75 studentů
Procentuální úspěšnost:	**57.3%**	53.2%
Předmět nedokončilo:	61 studentů	**66 studentů**
Procentuální neúspěšnost:	42.7%	**46.8%**
Nejlepší dosažené hodnocení:	**96.25b**	93.25b

Tabulka 14: Výsledek porovnání kurzů ML 2013/2014 a ML 2014/2015

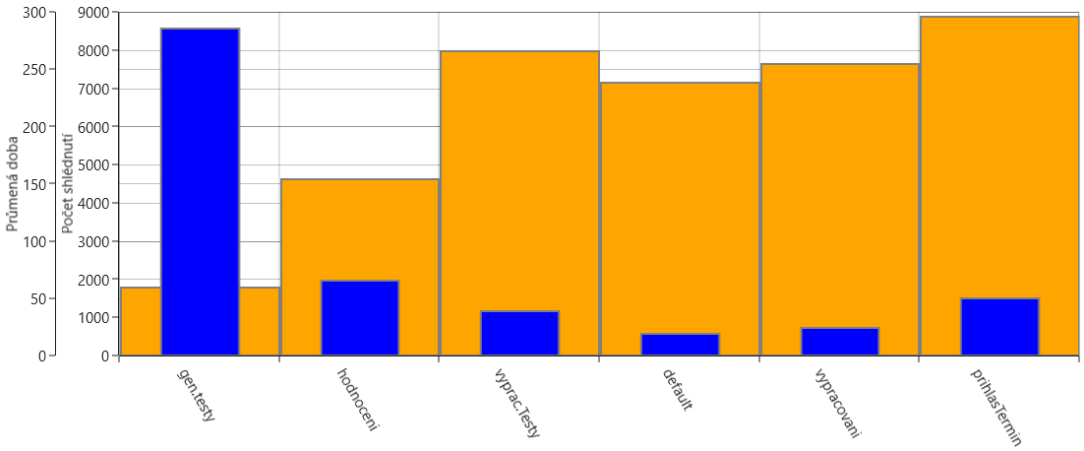
Příklad 2: Doporučené rozmístění prvků
Do statistické analýzy řadíme i doporučené rozmístění prvků. Používáme zde statistiky užívaní jednotlivých stránek v systému. Tyto statistiky počítáme z logů uživatelů. Dokážeme tak vyčíst jednotlivé četnosti zobrazení stránek a i průměrnou dobu zobrazení stránky.

Tyto informace promítneme do grafu a setřídíme vypočítané důležitosti. Nyní jako příklad uvedu Matematickou logiku za rok 2014/2015 prezenční studium.

Nastavené parametry:[9]

- Minimální % zobrazení aktivity - 6%
- Minimální % průměrné délky navštívení - 6%

Výsledné doporučení:



Obrázek 15: Graf - doporučené rozmístění prvků pro ML 2014/2015

Pro lepší orientaci, uvedeme i tabulku 15 doporučeného rozmístění prvků.

Název aktivity	Url	Průměrná doba	Počet shlédnutí
gen.testy	/Pages/Other/GeneratedTest.aspx	287.18s	1838
hodnoceni	/Pages/Student/HodnoceniKurzu.aspx	66.61s	4675
vyprac.testy	/Pages/Student/VypracovaneTesty.aspx	40.42s	8022
default	/Pages/Default.aspx	20.38s	7185
vypracovani	/Pages/Student/Vypracovani.aspx	25.25s	7684
prihlasTermin	/Pages/Student/PrihlaseniNaTermin.aspx	51.84s	8922

Tabulka 15: Doporučené rozmístění prvků pro ML 2014/2015

[9]Nastavenými parametry, odfiltrujeme aktivity, které nesplňují minimální četnosti.

7.1.1 Zhodnocení

Statistická analýza dat nám ukázala jednotlivé informace, které uživatel tohoto systému nemá k dispozici, jako například průměrné body za zkoušku/zápočet v analyzovaném kurzu. Všechny tyto informace na jednom místě, nám umožňují jednoduché porovnání s předchozími kurzy. Tyto výsledky může vyučující využít jako odrazový můstek, pro určovaní složitosti testů v následujících letech.

Vetší přínos má doporučování rozmístění prvků v systému. Významné jsou informace o používaní jednotlivých stránek, času stráveném na stránce a seřazení, které tato aplikace doporučuje.

Analýzou jsme zjistili, že uživatelé tráví nejvíce času na stránce pro generovaní testů. Tato stránka však nemá výrazný počet návštěv. Doporučíme tedy její posunutí na viditelnější místo v systému, aby byla lépe dostupná pro uživatele. Také hodnocení kurzu, je důležitá stránka, u které bychom měli zvážit její přesunutí na vhodnější místo. Analýzou doporučení rozmístění prvků, dokážeme sledovat používaní stránek v jednotlivých letech a adekvátně na tyto změny reagovat.

7.2 Výsledky shlukovací analýzy

Shlukovací analýza je rozdělena na analýzu kurzů, tříd a predikci výsledků kurzů. Za pomoci analýzy logů přístupů a výpočtu matice četností. Uvedeme si tedy příklad shlukové analýzy kurzů a predikce výsledků a zhodnotíme výsledky.

Příklad 1: Shlukovací analýza kurzů
Pro příklad uvedeme analýzu předmětu matematická logika za rok 2014/2015 prezenční studium.

Vstupní data:
Výčet četností všech aktivit (stránek) pro jednotlivé studenty v kurzu matematické logiky v roce 2014/2015 prezenčního studia. Ve vybraném kurzu se nachází celkem 30 aktivit, a 141 studentů.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	A_{13}	A_{14}	...	A_{30}
S_3	1143	122	54	52	13	2	99	12	29	45	80	4	3	1	...	0
S_{1043}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	...	0
S_{1211}	42	36	8	3	5	37	43	1	1	2	1	8	102	6	...	0
S_{1279}	6	3	6	9	2	1	1	1	5	0	0	0	0	0	...	0
S_{1301}	18	9	4	16	1	1	4	1	2	8	1	0	0	0	...	0
S_{1968}	138	92	8	5	78	106	30	3	5	15	7	3	1	9	...	0
S_{1978}	2	1	2	3	0	0	0	0	0	0	0	0	0	0	...	0
S_{1981}	26	17	2	3	5	2	20	23	24	2	9	10	1	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
S_{3353}	38	20	36	33	5	52	14	14	10	0	0	0	0	0	...	0

Obrázek 16: Matice četností aktivit pro ML 2014/2015

Výsledné četnosti aktivit, shlukujeme pomocí metody K-means. Pro zjištění počtu shluků aplikujeme Silhoulette index, u kterého dostaneme K rovno třem, takže data budeme rozdělovat do tří shluků.

Shluk	**Počet studentů**
C1	34
C2	160
C3	1

Tabulka 16: Shluková analýza kurzů - rozdělení do tří shluků

V clusteru C3 máme pouze jednoho studenta, jedná se tedy o odlehlé pozorování, pro přesnost analýzy odstraníme všechna odlehlá pozorování.

Shluk	**Počet studentů**
C1	71
C2	51
C3	11

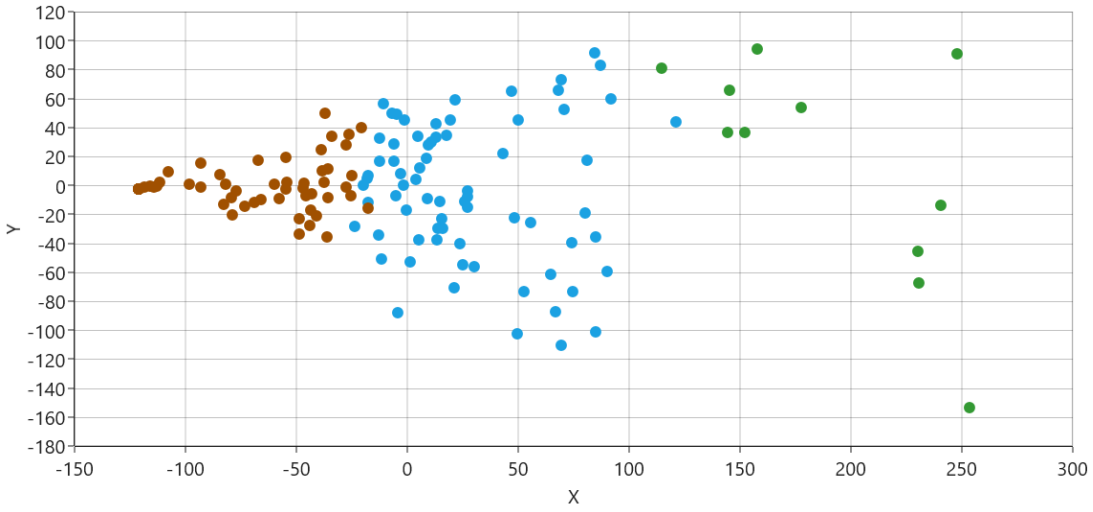
Tabulka 17: Shluková analýza kurzů - odstranění odlehlých pozorování

Pokud si budeme chtít jednotlivé studenty zobrazit do 2D grafu, použijeme PCA pro redukci dimenze a redukujeme tedy na dvě dimenze (viz kapitola 3.2.4).

	A_a	A_b
S_3	−975.6492818	378.7251922
S_{1043}	123.208646	31.97222221
S_{1211}	58.45651018	3.201894563
S_{1279}	114.29852	26.87545962
S_{1301}	100.0435063	25.80342717
S_{1968}	−57.39938575	−3.199266725
S_{1978}	120.542581	30.87375323
S_{1981}	86.06847018	21.02576885
S_{1995}	123.208646	31.97222221
S_{1997}	56.09861191	34.88405083
S_{1999}	123.208646	31.97222221
...	...	...
S_{3353}	65.53609457	3.881789479

Obrázek 17: PCA - výsledná data po redukci dimenze

Vezmeme-li výslednou matici (viz obr. 17) a zařadíme jednotlivé studenty do shluků a odlišíme shluky barvou, dostaneme graf (viz obr. 18).



Obrázek 18: Interpretace výsledků do 2D grafu

Příklad 2: Predikce výsledků kurzů

Podle roku 2013/2014 budeme tedy chtít predikovat rok 2014/2015. Pomocí měření vzdáleností (viz kapitola 3.2.6), najdeme nejmenší vzdálenost mezi jednotlivými studenty.

Rozdělení shluků pro rok 2013/2014 po odstranění odlehlých pozorování:

Shluk	Počet studentů	Úspěšnost	Průměrné body
C1	15	86.67%	67.51b
C2	40	85%	64.95b
C3	85	38.82%	33.32b

Tabulka 18: Predikce výsledků kurzů - rozdělení do tří shluků - ML 2013/2014

Z tabulky 18 si všimneme třetího shluku C3, u kterého zaznamenáváme nízké procento úspěšnosti. Jedná se tedy o 85 studentů, s podobným chováním v systému, kde pouze 32 studentů úspěšně dokončilo kurz.

Podíváme se tedy na jednotlivé studenty z následujícího roku 2014/2015, jestli jejich výsledky souhlasí s podobností v chování v minulém roce.

Pro shluk C3:
Studenti ve shluku C3 z tabulky 18, dosahovali průměrného hodnocení 33.32b a procentuální úspěšnosti 39%. Pomocí měření vzdálenosti, jsme nalezli 59 studentů v roce 2014/2015, kteří se svým chováním nejvíce přibližují zkoumané skupině C3 z roku 2013/2014. Ve výsledné tabulce 20 vidíme značnou podobnost jak v průměrném bodovém hodnocení studentů, tak i v procentuální úspěšnosti.

Počet studentů	Úspěšnost	Průměrné body
59	28.81%	29.68b

Tabulka 19: Predikce studentů v ML 2014/2015 pro shluk C3

7.2.1 Zhodnocení

Podle zjištěných výsledků analýz jsme schopni pomocí shlukové analýzy rozdělit studenty do skupin podle společného chování. Jednotlivé skupiny vykazují významné rozdíly v procentuální úspěšnosti studentů. Tento fakt nám pomáhá k odhalení typického chování studenta, který neuspěje v daném kurzu. Na základě těchto informací, můžeme predikovat, podle chování budoucích uživatelů, jestli úspěšně dokončí kurz nebo ne. Tyto informace mohou posloužit jako varovný signál studentům, kteří spadají do kategorie neúspěšných ještě před koncem kurzu.

7.3 Výsledky asociačních pravidel

Asociační pravidla nám tedy ukazují s jakou spolehlivostí nastanou dvě a více aktivit společně. U analýzy asociačními pravidly si také můžeme zvolit mezi analýzou kurzu nebo třídy. Jednotlivé skupiny uživatelů mají různé výsledné asociace a jejich spolehlivosti. Uvedeme si tedy příklad pro jeden specifický kurz, u kterého si zhodnotíme výsledek analýzy.

Příklad: Asociační pravidla kurzu
Pro příklad uvedeme analýzu předmětu matematická logika, který má nejvíce dostupných dat. Zvolíme si rok 2013/2014 a prezenční studium.

Nastavení minimální spolehlivosti a podpory:

Minimální podpora = 10%
Minimální spolehlivost = 60%

Antecedent	Sukcedent	Spolehlivost [%]
vyber_skoly & hodnoceni & vypracovani	default	100
vyber_skoly & prihlasTermin	default	99.97
vyber_skoly & vyprac.Testy	default	99.95
vyber_skoly & vypracovani	default	99.95
vyber_skoly & prihlasTermin & vyprac.Testy	default	99.95
default & aktivity	vyber_skoly	99.89
gen.testy	vypracovani	99.87

Tabulka 20: Asociační pravidla pro ML 2013/2014

Z této analýzy vidíme, že téměř každá transakce vede nejčastěji ke stránkám *default* a *vyber_skoly*. Tyto stránky nejsou vhodné pro analyzování jednotlivých transakcí, protože nemají žádnou vypovídající hodnotu. Každý uživatel si v systému vybírá školu hned po přihlášení, takže se nejedná přímo o aktivitu, kterou by chtěl uživatel provést, ale musí jí provést. Další aktivita je *default*, což je hlavní stránka následující hned po výběru školy. Pokud odfiltrujeme tyto dvě stránky z analyzovaných transakcí a provedeme analýzu znova dostaneme výsledky (viz tab. 21).

Antecedent	Sukcedent	Spolehlivost [%]
gen.testy	vypracovani	99.87
aktivity	prihlasTermin	85.48
hodnoceni & vypracovani	prihlasTermin	85.48
vyprac.Testy & vypracovani	prihlasTermin	80.28
vypracovani	prihlasTermin	77.24
hodnoceni & vyprac.Testy	prihlasTermin	73.86
hodnoceni & vypracovani	vyprac.Testy	67.81

Tabulka 21: Asociační pravidla pro ML 2013/2014 - filtrované

7.3.1 Zhodnocení

Součástí dobře nalezených asociačních pravidel, je tedy aplikování určitých filtrů. Obecně se jedná o stránky, u kterých víme, že nastávají v každé transakci (přihlašování, hlavní stránka atd.). Pokud z jednotlivých transakcí tyto stránky vypustíme, dokážeme nalézt v asociacích zajímavá pravidla. V uvedeném příkladu například vidíme, že s 99.9% spolehlivostí si student automaticky po vygenerovaní testu, přejde na jeho vypracování, což je poněkud logické. Další pravidlo říká, že hodnocení a zároveň vypracování implikuje přihlášení na termín s 80% spolehlivostí. Analýza asociací nám může sloužit jako pomocný nástroj pro doporučení rozmisťování prvků v systému eLogika.

8 Závěr

V práci jsme uvedli čtenáře do problematiky analýzy dat pomocí dobývaní znalostí z databází a jejich individuálních kroků. Následně jsme se věnovali popisu jednotlivých data miningových metod, které jsem použil pro realizaci této aplikace. Podrobněji jsme si analyzovali logy přístupů uživatelů systému eLogika a uvedli jsme si dva návrhy dodatečných dat k logování. Prvním návrhem bylo logování aktivit v mobilní aplikaci systému eLogika. Dalším návrhem je zavedení session managementu, který by zajišťoval přehledné oddělení jednotlivých logů. Tyto doplňující data nám umožní vylepšit vypovídající hodnotu analýzy chování uživatelů v systému. Popsali jsme si metody K-means, PCA, algoritmus Apriori, které jsme si naimplementovali. Součástí implementace bylo rozdělení aplikace do tří částí na aplikační, databázovou a prezentační. Tyto části jsou od sebe odděleny, aby splňovaly požadavek znovupoužitelnosti aplikace. Aplikační a databázová část byly navrženy jako PCL knihovny, které se dají přenášet napříč platformami. Ostatní knihovny nezbytné pro vývoj aplikace, byly uvedeny v seznamu použitých knihoven.

Tato práce vyžadovala detailnější studium a pochopení vnitřní databázové struktury systému eLogika. Rozpoznali jsme, jaká data systém poskytuje a popsali jsme si ta, která budeme používat pro výslednou analýzu.

Analýzou chování studentů v systému jsme zjistili, že rozdělením studentů do skupin podle chování, se jednotlivé skupiny značně odlišují procentuální úspěšností a průměrným bodovým hodnocením. Tento fakt nám odhalil skutečnost, že chování studentů v systému má dopad na jejich úspěšnost v zapsaném kurzu. Aplikovali jsme tedy výsledky dat z kurzů v předešlých letech na kurzy v letech následujících. Ověřili jsme, že u některých již skončených kurzů, dokážeme predikovat výsledky studentů na základě analýzy z předešlých ročníků. Toto tvrzení nešlo ověřit u všech skončených kurzů, z důvodu absence logovaných dat v některých kurzech. Navrhli jsme použít výsledky z minulosti pro predikci chování studentů v aktuálních kurzech a doporučit studentům s rizikovým chováním změnu přístupu ke studiu.

Při práci jsme si rozšířili vědomosti v oblasti analýzy vícerozměrných dat a hlavně data miningu. Prošli jsme si celým vývojovým procesem univerzální aplikace pro Windows 10, jejímž výsledkem je úspěšné nasazení samotné aplikace na Windows Store. Z těchto důvodu se práce jeví jako přínosná pro autora i pro systém e Logika.

Jakub Gereg

Literatura

[1] HANCOCK, Monte. Practical data mining. New York: CRC Press, c2012. ISBN 978-1-4398-6836-2.

[2] HAN, Jiawei, Micheline KAMBER a Jian PEI. Data mining: concepts and techniques. 3rd ed. Boston: Elsevier, c2012. Morgan Kaufmann series in data management systems. ISBN 978-0-12-381479-1.

[3] CIOS, Krzysztof. Data mining: a knowledge discovery approach. New York: Springer Sience+Business Media, 2007. ISBN 978-0-387-33333-5.

[4] BERKA, Petr. Dobývání znalostí z databází. Vyd. 1. Praha: Academia, 2003. ISBN 80-200-1062-9.

[5] RUD, Olivia Parr. Data Mining: praktický pruvodce dolováním dat pro efektivní prodej, cílený marketing a podporu zákazníku (CRM). Praha: Computer Press, 2001. Databáze. ISBN 80-7226-577-6.

[6] ROMERO, Cristóbal a Sebastián VENTURA. Educational Data Mining: A Review of the State of the Art. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) [online]. 2010, 40(6), 601-618 [cit. 2016-04-17]. DOI: 10.1109/TSMCC.2010.2053532. ISSN 1094-6977. Dostupné z: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5524021`

[7] KOB, Hian Chye a Gerald TAN. Data Mining Applications in Healthcare. Journal of Healthcare Information Management [online]. , 9 [cit. 2016-04-19]. Dostupné z: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.3184&rep=rep1&type=pdf`

[8] NADAF, Mohsin a Vidya KADAM. Data Mining in Telecommunication [online]. 2013, (2) [cit. 2016-04-19]. ISSN 2319–2526. Dostupné z: `http://www.irdindia.in/journal_ijacte/pdf/vol2_iss3/16.pdf`

[9] ANDEL, Jirí. Statistické metody. 2. preprac. vyd. Praha: Matfyzpress, 1998. ISBN 80-85863-27-8.

[10] EVERITT, Brian. Cluster Analysis. 5th ed. Chichester: Wiley, 2011. Wiley series in probability and mathematical statistics, 848. ISBN 978-0-470-74991-3.

[11] HARUŠTIAKOVÁ, Danka. Vícerozměrné statistické metody v biologii. Vyd. 1. Brno: Akademické nakladatelství CERM, 2012. ISBN 978-80-7204-791-8.

[12] SMITH, Lindsay I. A tutorial on Principal Components Analysis [online]. 2002, 26 [cit. 2016-04-17]. Dostupné z: `http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf`

[13] STEINLEY, Douglas. K-means clustering: A half-century synthesis. British Journal of Mathematical and Statistical Psychology [online]. 2006, 59(1), 1-34 [cit. 2016-04-17]. DOI: 10.1348/000711005X48266. ISSN 00071102. Dostupné z: http://doi.wiley.com/10.1348/000711005X48266

[14] DEZA, M a Elena DEZA. Encyclopedia of distances. New York: Springer Verlag, c2009. ISBN 9783642002342.

[15] TAN, Pang-Ning, Michael STEINBACH a Vipin KUMAR. Introduction to data mining. Boston: Pearson Addison Wesley, c2006. ISBN 0-321-32136-7.

[16] Data-mining nad systémem eLogika. Kognitivní věda a umělý život. V Opavě: Slezská univerzita, Filozoficko-přírodovědecká fakulta, Ústav informatiky, 2013, s. 181-187. ISBN 978-80-7248-863-6.

[17] FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures [online]. Irvine, 2000 [cit. 2016-04-17]. Dostupné z: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf. Doctoral dissertation. University of California.

[18] Pre Processing of Web Logs – An Improved Approach For E-Commerce Websites [online]. 2015 [cit. 2016-04-17]. ISSN 0975-4024. Dostupné z: http://www.enggjournals.com/ijet/docs/IJET15-07-01-328.pdf

[19] ČUBOŇ, Mário. ELogika nad platformou Windows 8 Windows Phone 8. 2015. VŠB – Technická univerzita Ostrava Fakulta elektrotechniky a informatiky Katedra informatiky. Vedoucí práce Mgr. Marek Menšík, Ph.D.

[20] BROWN, Pete. Windows store app development: C# and XAML. Shelter Island, N.Y.: Manning, 2013. ISBN 1617290947.

[21] TAN, Pang-Ning, Michael STEINBACH a Vipin KUMAR. Introduction to data mining. Boston: Pearson Addison Wesley, c2006. ISBN 0-321-32136-7.

[22] ZAKI, Mohammed J a Wagner MEIRA. Data mining and analysis: fundamental concepts and algorithms. New York, NY: Cambridge University Press, 2014. ISBN 9780521766333.

[23] Get started with Windows apps: Guide to Universal Windows Platform (UWP) apps. Microsoft Developer Network [online]. 2016 [cit. 2016-04-17]. Dostupné z: https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

[24] Get started with Windows apps: App lifecycle. Microsoft Developer Network [online]. 2016 [cit. 2016-04-17]. Dostupné z: https://msdn.microsoft.com/en-us/windows/uwp/launch-resume/app-lifecycle

[25] Developing for Multiple Platforms with the .NET Framework: Cross-Platform Development with the Portable Class Library. Microsoft Developer Network [online]. 2016 [cit. 2016-04-17]. Dostupné z: https://msdn.microsoft.com/en-gb/library/gg597391(v=vs.110).aspx

[26] MACVITTIE, Lori A. XAML in a nutshell. 1st ed. Sebastopol, CA: O'Reilly, 2006. In a nutshell (O'Reilly & Associates). ISBN 9780596526733.

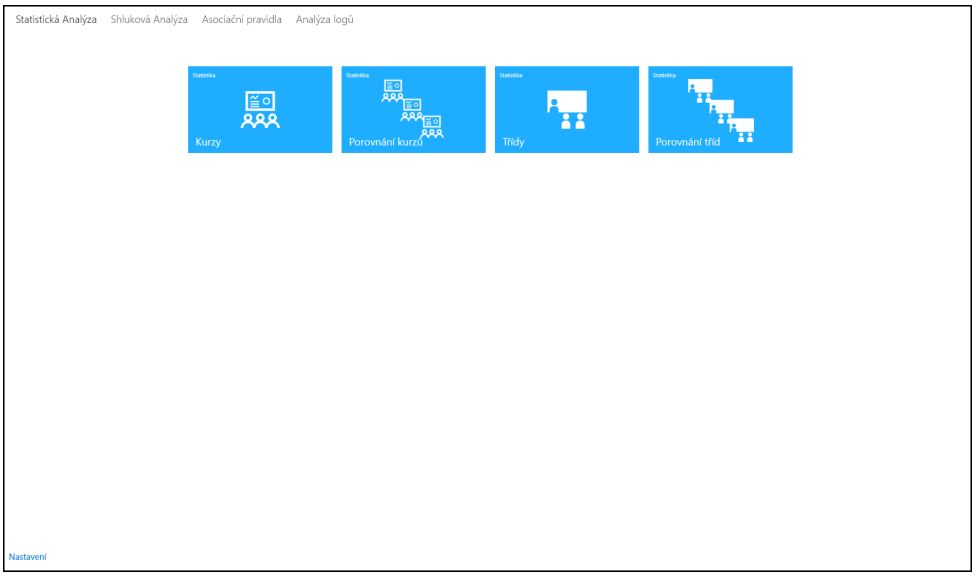
[27] The MVVM Pattern. Microsoft Developer Network [online]. 2016 [cit. 2016-04-17]. Dostupné z: https://msdn.microsoft.com/en-us/library/hh848246.aspx

[28] Implementing the MVVM Pattern Using the Prism Library 5.0 for WPF. Microsoft Developer Network [online]. 2016 [cit. 2016-04-17]. Dostupné z: https://msdn.microsoft.com/en-us/library/gg405484(v=pandp.40).aspx

A Uživatelská příručka

Tato příručka vás seznámí s funkcemi analytického nástroje pro systém eLogika.

A.1 Menu statistická analýza

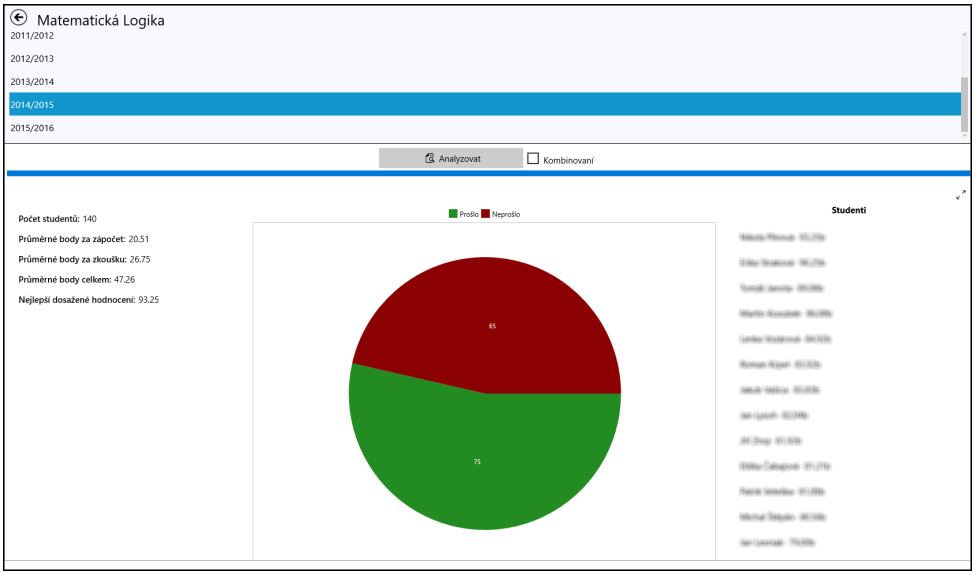


Obrázek 19: Menu statistická analýza

Statistická analýza obsahuje tyto možnosti:

- Kurzy
- Porovnání kurzů
- Třídy
- Porovnání tříd

Kurzy:
V horním menu si zvolíme kurz, který chceme analyzovat. V pořadí volby (školy – předmětu – akademického roku). Stiskneme tlačítko „Analyzovat“.

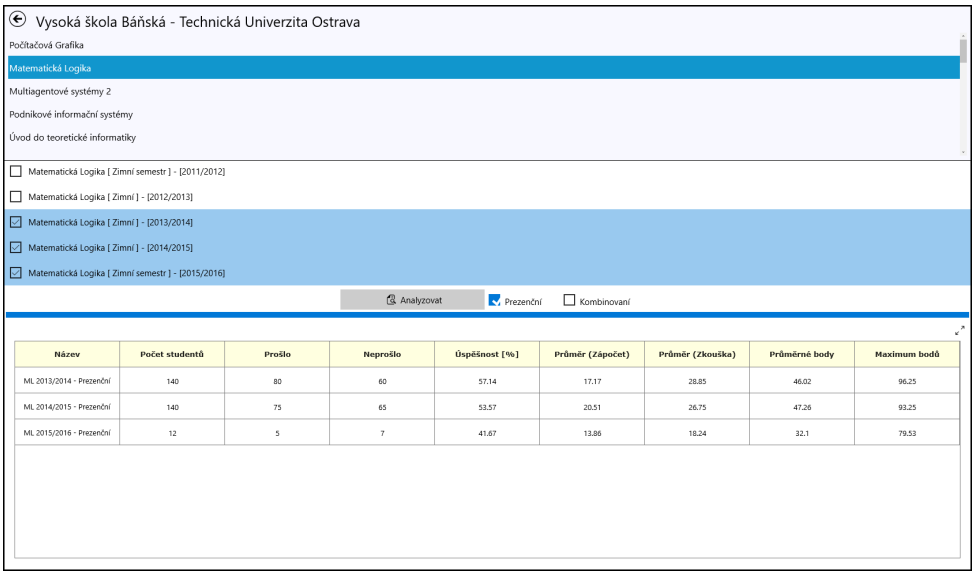


Obrázek 20: Statistická analýza - kurzy

Porovnání kurzů:

Porovnání kurzů obsahuje horní menu, ve kterém si můžeme vybrat analyzované kurzy. Vybíráme nejprve školu, ve které se kurz nachází, následovně pak předmět. Pomocí zaškrtávacího seznamu zvolíme individuální kurzy a stiskneme tlačítko „Analyzovat“.

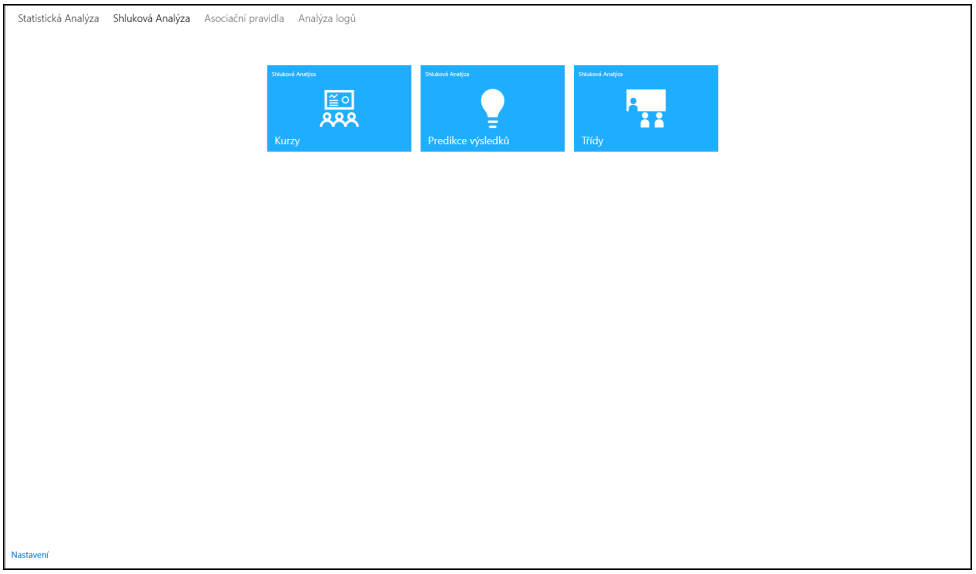
Ve výchozím stavu je nastavená analýza pouze pro kurzy prezenčního studia, pomocí zaškrtávacího políčka si můžeme zvolit i analýzu kurzů kombinovaného studia.



Obrázek 21: Statistická analýza - porovnání kurzů

Obdobný způsob ovládání, nalezneme při analýze a porovnávání tříd.

A.2 Menu shluková analýza



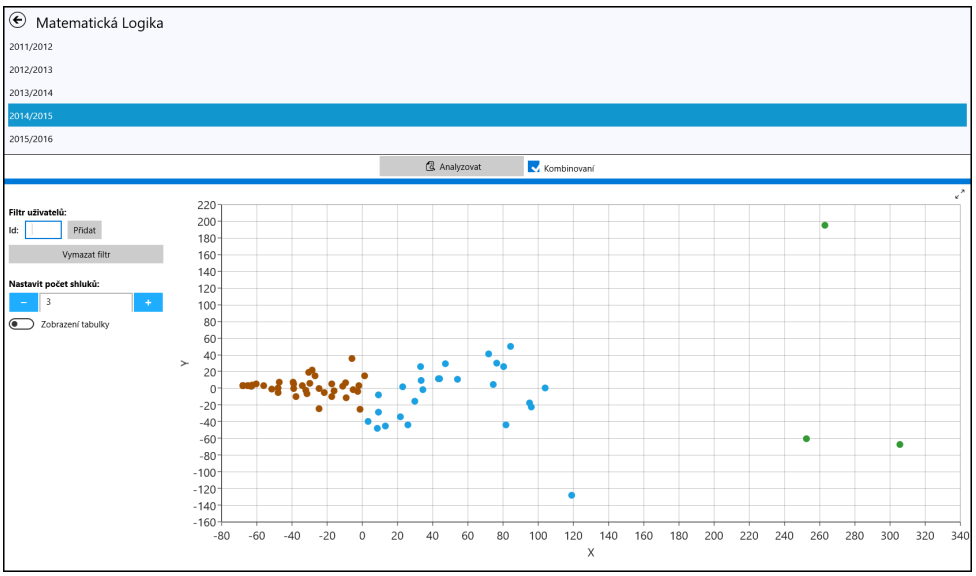
Obrázek 22: Menu shluková analýza

Shluková analýza obsahuje tyto možnosti:

- Kurzy
- Predikce výsledků
- Třídy

Kurzy:
V horním menu si zvolíme kurz, který chceme analyzovat. V pořadí volby (školy – předmětu – akademického roku). Stiskneme tlačítko „Analyzovat“.

Po provedení analýzy si můžeme výsledná data upravovat pomocí parametrů. Můžeme si tak přepínat mezi zobrazením grafu a tabulky, nastavovat si počet shluků a filtrovat si odlehlá pozorování.

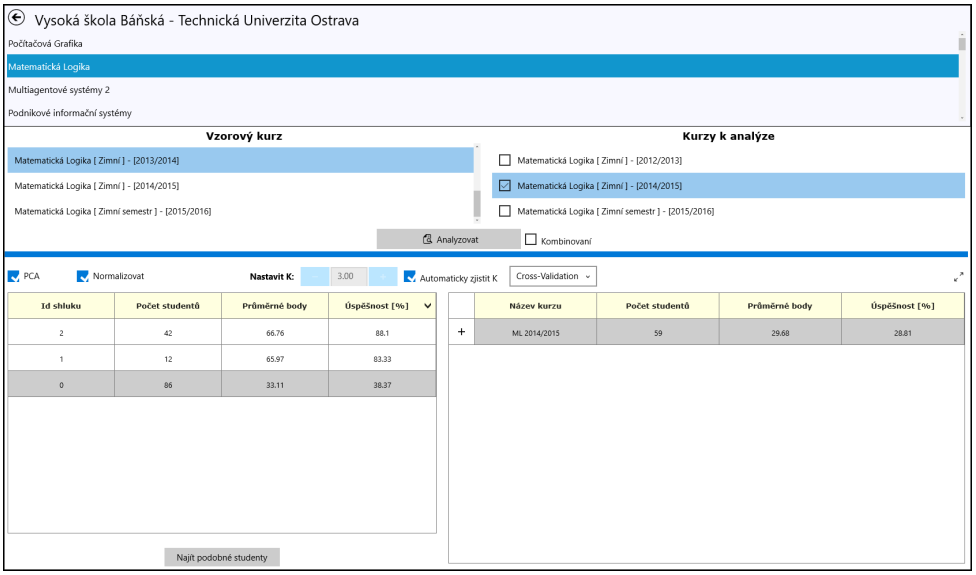


Obrázek 23: Shluková analýza - kurzy

Obdobný způsob ovládání, nalezneme při analýze tříd.

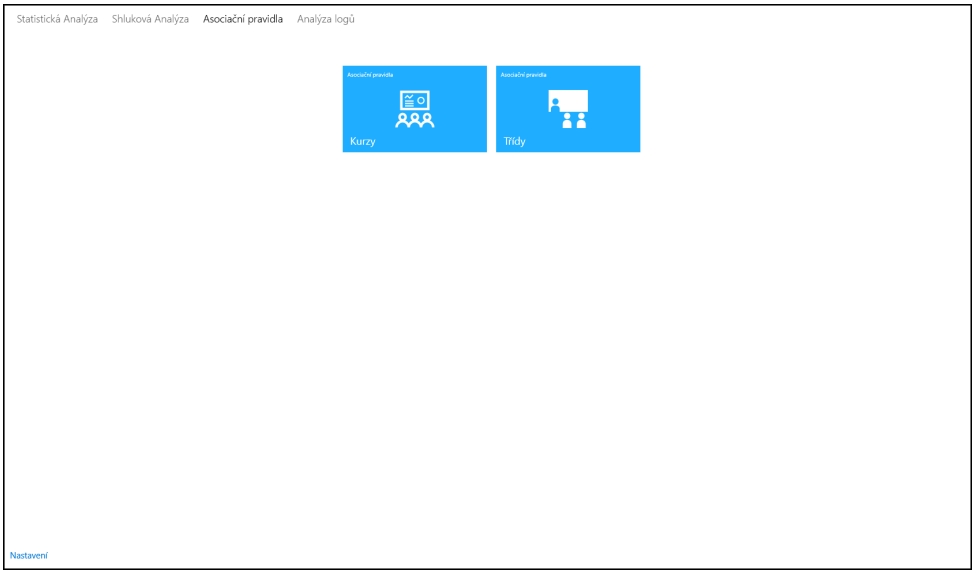
Predikce výsledků:
V horním menu si zvolíme předmět, který chceme analyzovat. Nastavíme si jeden vzorový kurz, a zvolíme si kurzy, u kterých budeme provádět predikci. Stiskneme tlačítko „Analyzovat“. Po načtení dat si můžeme zvolit ze čtyř parametrů (PCA, normalizace, nastavení počtu shluků, automatické zjišťování shluků).

V levé části si vybereme shluk, který chceme použít jako vzor pro nalezení podobných uživatelů ve zvolených kurzech. Stiskneme tlačítko „Najít podobné studenty“. V pravé tabulce se nám zobrazí skupiny uživatelů ve zvolených kurzech, kteří mají podobné chování. Porovnáním výsledných hodnot a procentuálních úspěšností, můžeme určit, zda predikce dopadla úspěšně.



Obrázek 24: Shluková analýza - predikce výsledků

A.3 Menu asociační pravidla



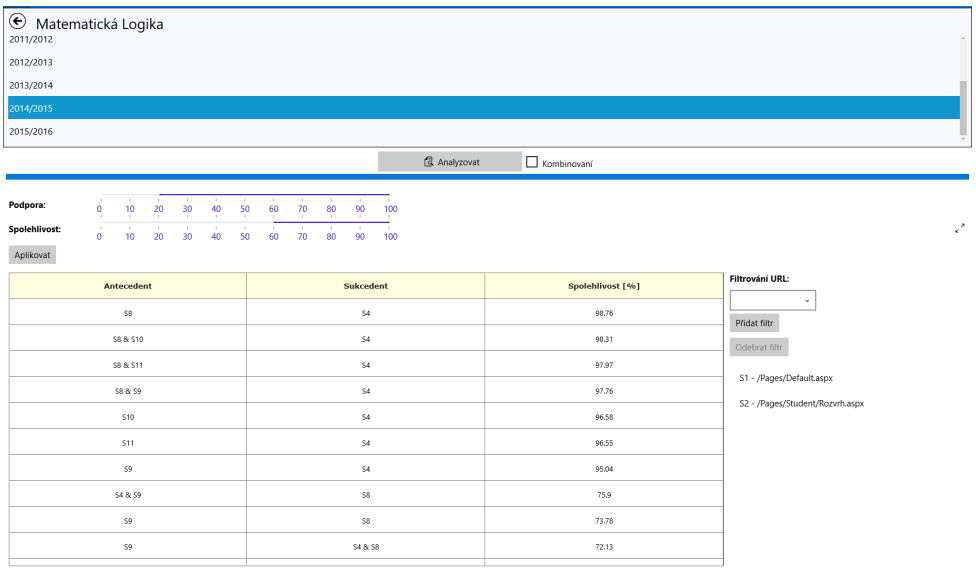
Obrázek 25: Menu asociační pravidla

Asociační pravidla obsahují tyto možnosti:

- Kurzy
- Třídy

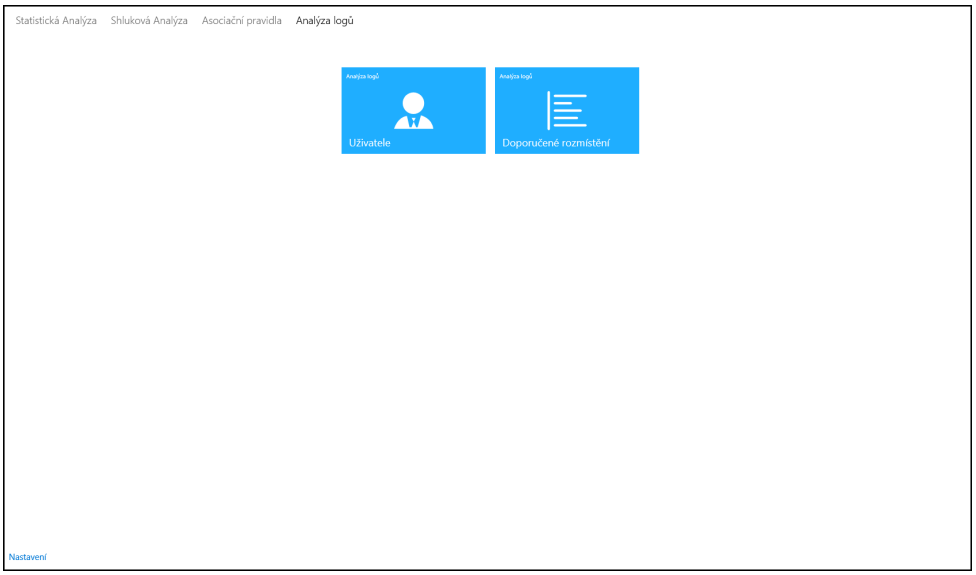
Kurzy:
V horním menu si zvolíme kurz, který chceme analyzovat. V pořadí volby (školy – předmětu – akademického roku). Stiskneme tlačítko „Analyzovat“. Po provedení analýzy si zvolíme požadovanou minimální podporu a spolehlivost. Stiskneme tlačítko „Aplikovat“.

Na výsledná pravidla, můžeme také aplikovat filtr, který nám z transakcí odstraní zvolené aktivity.



Obrázek 26: Asociační pravidla - kurzy

A.4 Menu analýza logů

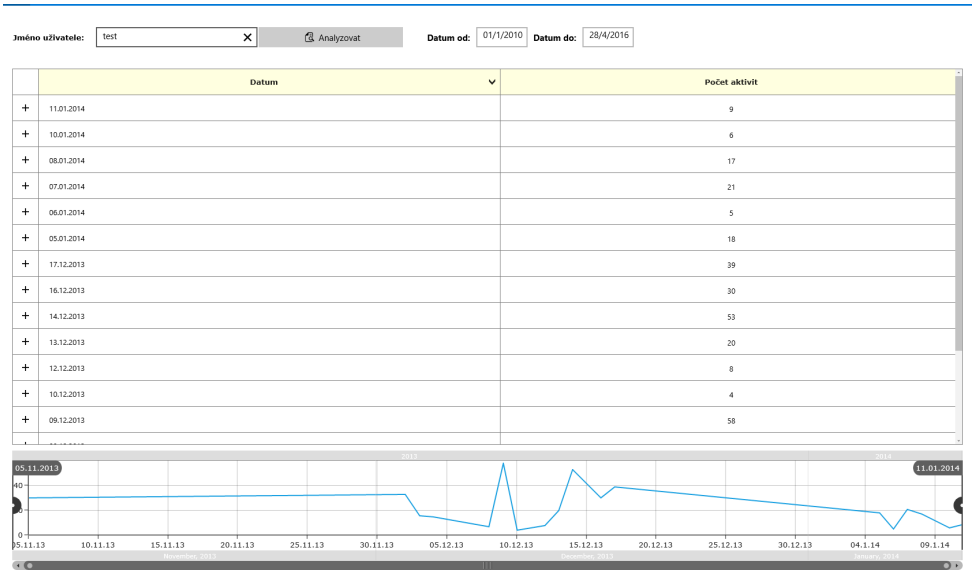


Obrázek 27: Menu analýza logů

Analýza logů obsahuje tyto možnosti:

- Uživatele
- Doporučené rozmístění

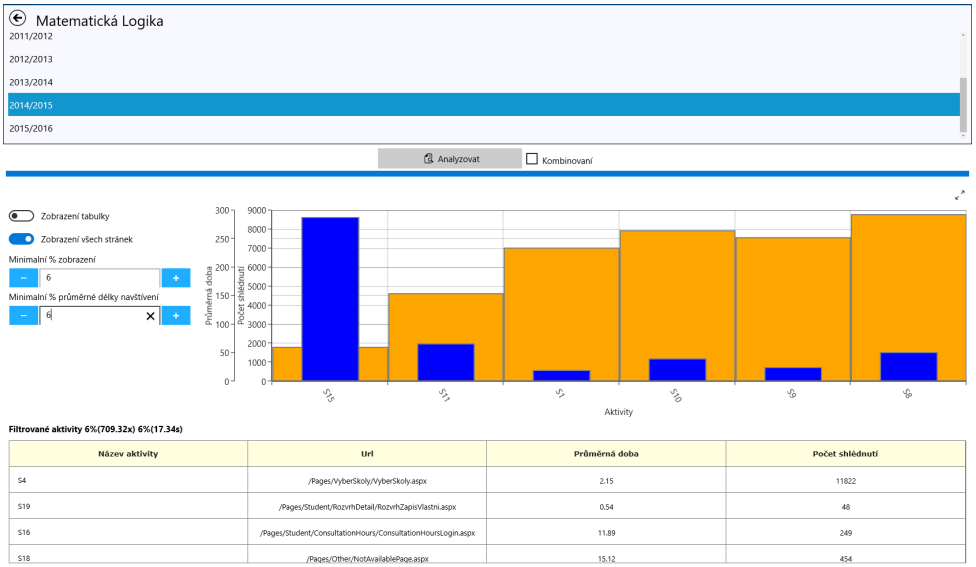
Uživatele:
Zde vyplníme jméno nebo školní login uživatele, kterého chceme analyzovat. Nastavíme interval zobrazení aktivit v systému eLogika (od – do). Stiskneme tlačítko „Analyzovat“. Zobrazí se nám detailní graf a tabulka jednotlivých dnů, u kterých si můžeme rozbalit obsah logů.



Obrázek 28: Analýza logů - uživatele

Doporučené rozmístění:
V horním menu si zvolíme kurz, který chceme analyzovat. V pořadí volby (školy – předmětu – akademického roku). Stiskneme tlačítko „Analyzovat“.

Po načtení analyzovaných dat, můžeme upravovat pět parametrů. Prvním parametrem je přepínaní mezi zobrazováním tabulky a grafu. Dále pak volba mezi zobrazením všech stránek, nebo pouze doporučených. Filtrovat aktivity můžeme podle nastavení minimálního procenta zobrazení a minimální délky navštívení stránky.



Filtrované aktivity 6%(709.32x) 6%(17.34s)

Název aktivity	Url	Průměrná doba	Počet shlédnutí
S4	/Pages/VyberSkoly/VyberSkoly.aspx	2.15	11822
S19	/Pages/Student/RozvrhDetail/RozvrhZapisVlastni.aspx	0.54	48
S16	/Pages/Student/ConsultationHours/ConsultationHoursLogin.aspx	11.89	249
S18	/Pages/Other/NotAvailablePage.aspx	15.12	454

Obrázek 29: Analýza logů - doporučené rozmístění

B Obsah přiloženého CD

Přiložené CD obsahuje tyto složky:

- *Aplikace* - obsahuje projekt aplikace ve Visual Studiu 2015
- *Diplomová práce* - obsahuje text diplomové práce ve formátu .pdf